PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-147800

(43)Date of publication of application : 29.05.2001

(51)Int.Cl. G06F 7/24

G06F 12/00

G06F 12/06

G06F 15/16

G06F 17/30

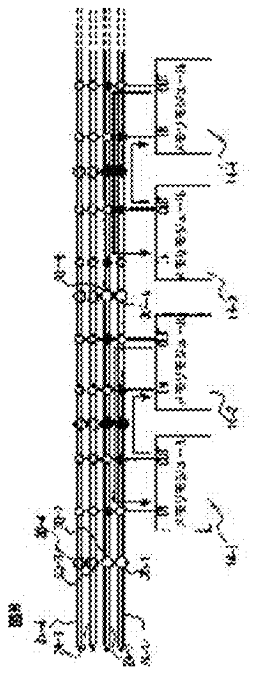
(21)Application number : 11-330997

(22)Date of filing : 22.11.1999

(71)Applicant : TAABO DATA LABORATORY KK

(72)Inventor : KOSHO SHINJI

(54) INFORMATION PROCESSING SYSTEM AND SORTING METHOD, COMPILING METHOD AND JOINING METHOD UTILIZING THE INFORMATION PROCESSING SYSTEM



(57)Abstract:

PROBLEM TO BE SOLVED: To perform sorting of an array or the like in stable processing time at a markedly high speed.

SOLUTION: Relating to a distributed memory type information processor, presentation memory modules 14-1 and 14-3 supplies elements sorted inside their own memory modules, together with order numbers via a bus 24 divided by a switch 30 or the like to judgment memory modules 14-2 and 14-4. The judgment memory modules calculate a virtual order number for indicating the candidate of the order number of the received element and returns the virtual order number via the other bus 24 to the presentation memory modules. When receiving the virtual order number, the presentation memory modules update the order number of the element, in accordance with the virtual order number.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

CLAIMS

[Claim(s)]
[Claim 1] The connection between a CPU module, two or more memory modules with which each has MPU and a RAM core, and said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU Are the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module, and the sort of the element with which MPU of said memory module constitutes the part of the array which self grasps is performed. The location which said part which said self grasps as the sort means which rearranges said element according to specific sequence occupies during an array is followed. Said sorted element with the ranking number It transmits to other memory modules through a predetermined bus. By the comparison with I/O which receives other said elements and ranking numbers from a memory module through a predetermined bus, and the element which self grasps when said element and a ranking number are received A ranking number calculation means to compute the virtual ranking number which is the candidate of the ranking number of the received element, and to return a memory module besides the above, The presentation memory module which is a memory module of the side which is equipped with a ranking decision means to decide the ranking of an element, according to the virtual ranking number concerned, and sends out said element and a ranking number when said virtual

ranking number is received, Information processing system characterized by deciding the ranking number of said array element by the communication link with the judgment memory module which are said element and a near memory module which computes a virtual ranking number by carrying out ranking number acceptance.
[Claim 2] An element specification / sending-out means to specify the element with which said memory module serves as a processing object according to the settled ranking number, and to send out to which bus, When the same element as an element comparison means to compare the element used as the last processing object with the sent-out element is sent out It has the same value number counter which counts up the value and which shows the number of existence of the same element. When it is judged that the element with which said element comparison means became the last processing object differs from the sent-out element It is constituted so that the value of the same value number counter about the element used as the last processing object and the element concerned may be associated and any may send out. Furthermore, information processing system according to claim 1 characterized by having the array arranged in the sequence which received the value of the element with which which memory module became the sent-out last processing object, and a related counter, and associated and received these.
[Claim 3] The value number counter with which said memory module shows the ranking number without duplication which counts up the value when it is judged that the element with which said element comparison means became the last processing object differs from the sent-out element, When the element used as the last processing object and the sent-out element are the same, about the sent-out element the value of a value number counter It is decided that it will be the ranking number of the element concerned without duplication. On the other hand, when [that] these differ The information processor according to claim 2 characterized by having determined the value of the counted-up value number counter as the ranking number of the element concerned without duplication, and having a renewal means of a ranking number to update the ranking number concerned.
[Claim 4] The connection between a CPU module, two or more memory modules with which each has MPU and a RAM core, and said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU Are the sort approach of an array of having used the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each

memory module, and it sets to the (a) memory module. The step which sorts the element which constitutes the part of the array which self grasps, (b) The location which said part which said self grasps occupies during an array is followed. The step which determines the judgment memory module of the side which receives the presentation memory module, element, and ranking number of the side which sends out an element and a ranking number among the memory modules which grasp the part of said array, (c) In a presentation memory module, the sorted element with the ranking number In the step transmitted to other memory modules through a predetermined bus, and (d) judging memory module In the step which receives other said elements and ranking numbers from a memory module through a predetermined bus, and the (e) aforementioned judging memory module Based on the ranking number of the element which the judgment memory module concerned grasps, the virtual ranking number which shows the candidate of the ranking number of the received element is computed. In the step which returns the virtual ranking number concerned to said presentation memory module, and the (f) aforementioned presentation memory module The step which updates the ranking number of an element according to the virtual ranking number concerned when said virtual ranking number is received, (g) whenever said step (d) - (f) is completed, the memory module group which consists of the presentation memory module and judgment memory module about the element to which the predetermined ranking number was given by concerned step (d) - (f) By repeating step (d) - (f) as either a presentation memory module group or a judgment memory module group, respectively The sort approach characterized by deciding the ranking number of each element of an array by updating the ranking number of the element in each memory module group.
[Claim 5] The sort approach according to claim 4 characterized by including the step to which said step (e) computes a virtual ranking number based on the number of front insertion which shows the number of the elements which should be located ahead, the ranking number about the element which should be located ahead, and the received ranking number from the element received (e1).
[Claim 6] The sort approach according to claim 4 or 5 characterized by said step (f) containing the step which substitutes the virtual ranking number received (f1) for the ranking number of the element sent out at the step (c).
[Claim 7] Furthermore, the element grasped in (h) presentation memory module group with the memory module which constitutes the presentation memory module group concerned So that it may have the step which computes the number of duplications which shows whether it has set in the memory module group concerned, and shoes

existence is recognized and it may not transmit [said step (c) overlaps the same (c1) element and] it The number of front insertion which shows the number of the elements which should be ahead located from the element with which said step (e) received the sorted element with the ranking number and the number of duplications including the step transmitted to other memory modules (e2), The step which computes a virtual ranking number is included based on the ranking number about the element which should be located ahead, and the ranking number and the number of duplications which were received. Said step (f) A virtual (f2) ranking number, The sort approach according to claim 4 characterized by including the step which determines the ranking number of the same element as the element concerned based on a difference with the ranking number at the time of sending out of the element in a step (c).
[Claim 8] The sort approach given in claim 4 characterized by forming the presentation memory module group which consists of a memory module which is 2n to which the increment of the n (n:1 or more integers) is carried out whenever a presentation memory module is an independent memory module, and a receiving module is also an independent memory module and the step of (d) - (f) ends it, and the judgment memory module group which consists of a 2n memory module in first stage thru/or any 1 term of 7.
[Claim 9] An array is sorted by the approach indicated by claim 4 thru/or any 1 term of 6. And based on the sorted array concerned, are the compile approach that the element under said array generates the new array arranged in predetermined sequence that there is no duplication, and it sets to a (i) predetermined memory module. When the same element as the step which sends out the element which serves as a processing object according to a ranking number, and the element used as the processing object of (j) last time is sent out Count up the same value number counter which shows the number of existence of the same element, and on the other hand, when the element used as the last processing object and a different element are sent out The step which associates the value of the same value number counter about the element used as the last processing object, and the element concerned, and sends these out, (k) by receiving the value of the element used as the last processing object, and the same related value counter, having the step which associates these and is arranged during a new array, and repeating (l) step (i) - (j) The compile approach characterized by associating and arranging an element and its number of existence during said new array.
[Claim 10] furthermore, (m) -- the compile approach according to claim 9

characterized by having the step which acts as the monitor of the value of the element sent out at a step (j), and the same related value number counter in which module, and performing a step (k) with which the module concerned.
[Claim 11] (n) While forming the ranking number counter and the same value number counter which store the ranking number of the element used as a processing object, and the number of existence of the element concerned in the memory module which grasps the array element concerned, respectively And it sets to the memory module which grasps the step which prepares the register which stores the element used as the last processing object temporarily, and the element to which the ranking number concerned was given according to (o) ranking number. When the step which sends out the element concerned to the 1st bus, and the element received in the memory module which grasps the (p) array element are compared with the content of the register and these are in agreement While counting up the number of existence, when these are not in agreement In which memory module the step which updates the content of the register, and the value of the number counter of existence after sending out the content of the register, and the value of the number counter of existence to the 2nd bus, and (q) -- The compile approach according to claim 9 characterized by having the step which arranges the content of said register, and the value of the number counter of existence during an array as the number of existence of an element and the element concerned, respectively.
[Claim 12] A step (n) is further related with the element used as a processing object (n1). Said step (p) compares with the content of the register the element received (p1) including the step which forms the value counter which stores a ranking number without duplication. When these are in agreement, while giving the value of a value number counter to the ranking number of the element used as the processing object concerned The compile approach according to claim 11 characterized by including the step which gives the value of the value number counter which counted up the value number counter and was counted up for the ranking number of the element used as a processing object when these are not in agreement.
[Claim 13] the sort approach given in any 1 term of claim 4 thru/or claim 8 -- and Are the join approach of an array of realizing share-ization of two or more arrays using the compile approach of a publication in claim 9 thru/or any 1 term of 12, and the array of (r) plurality is joined. The step which performs processing concerning said sort approach which gives a ranking number to each of these array elements, (s) The join approach of the array characterized by having the step which generates the new array in which processing concerning said compile approach is performed and the duplicate

element does not exist according to the element and its ranking number under said array which joined.
[Claim 14] The connection between a CPU module, two or more memory modules with which each has MPU and a RAM core, and said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU It is the join approach of two or more arrays of having used the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module. In the join approach of an array of realizing share-ization of two or more arrays using the compile approach of a publication in the sort approach given in any 1 term of claim 4 thru/or claim 8 and claim 9 thru/or any 1 term of 12 In order to specify the predetermined element in the value list said whose memory module is the array which stored the element based on the record number, respectively Equip the location corresponding to a record number with the pointer array which has arranged the pointer value which shows a value list, and two or more (r1) value lists are joined. The step which performs processing concerning said sort approach which gives a ranking number to each of these array elements, (t) While generating the new value list with which processing concerning said compile approach is performed, and the duplicate element does not exist according to the element and its ranking number under said value list which joined The step which updates the ranking number of said element for the ranking number of the element concerned in case the duplicate element does not exist, (u) The join approach characterized by having the step which considers the array which consists of a ranking number of an element in case said duplicate element does not exist as a new pointer array to show a new value list.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application] This invention relates to a detail extremely more about the information processor of a distributed memory type at the information processor

which can realize sort, compile, and processing of a join at a high speed.
[0002]
[Description of the Prior Art] A computer is introduced into various locations of the whole society, and large-scale data came to be stored here [there] by the end of today when networks including the Internet permeated. In order to process such large-scale data, huge count is required, therefore trying to introduce parallel processing is natural.
[0003] Now, parallel processing architecture is divided roughly into a "shared memory mold" and a "distributed memory type." The former ("shared memory mold") is a method with which two or more processors share one huge room. Since the traffic between a processor group and a shared memory serves as a bottleneck by this method, it is not easy to build a realistic system using the processor exceeding 100. In case the square root of 1 billion floating point variables is calculated by following, the acceleration ratio to a single CPU will call it at most 100 times. Experientially, about 30 times is an upper limit.
[0004] Each processor has respectively local memory, and the latter ("distributed memory type") combines these, and builds a system. The design of the hardware system which also incorporated hundreds - tens of thousands of processors by this method is possible. Therefore, it is possible to make the acceleration ratio to the single CPU at the time of calculating the square root of the 1 billion above-mentioned floating point variables one 10,000 times the number [hundreds -] of this.
[0005]
[Problem(s) to be Solved by the Invention] Although it is said that the potential need of the parallel processing by the processor of a large number exceeding hundreds is large, if a current realistic hardware technique tends to realize this as mentioned above, the design by technique other than a distributed memory type is difficult. In a distributed memory type, since the capacity of the memory attached to each processor is small, in the maintenance and processing of large-scale data (usually array) which are one of the main objects of parallel processing, the memory attached to two or more processors and each needs to allot this.
[0006] However, when the memory attached to two or more processors and each allots an array, the bus mediation for preventing the collision of the data on a bus is difficult, if each processor cannot operate in juxtaposition, utilization effectiveness of a processor cannot be improved, consequently there is a trouble of being unable to attain improvement in the speed of processing. Then, this invention attains the various objects as follows.

[0007] (1) The collision of the data on a bus does not occur in algorithm, but bus mediation is unnecessary and, thereby, raise processing speed taking advantage of the bandwidth of a bus fully.
(2) Enable it to assign the processing which made parallel processing by these possible, and used each memory module effectively combining many memory modules equipped with the processor (two or more desirable processors) and the graduation, and became independent to the processor in each memory module, and, thereby, raise processing speed further by deployment of a memory module.
(3) When magnitude of the data for a sort is set to "N", need only the magnitude of the data of O (N). (In the conventional sorting application, when the worst, the amount of data of O (N*N) or O (N*Log (N)) may be needed.)
(4) The processing time is stable, and even when the worst, expectable processing speed is guaranteed. That is, remarkably, this invention is the stable processing time and aims high-speed at offering the information processor which can sort an array.
[0008]
[Means for Solving the Problem] The connection between two or more memory modules with which, as for the object of this invention, a CPU module and each have MPU and a RAM core, and said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU Are the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module, and the sort of the element with which MPU of said memory module constitutes the part of the array which self grasps is performed. The location which said part which said self grasps as the sort means which rearranges said element according to specific sequence occupies during an array is followed. Said sorted element with the ranking number It transmits to other memory modules through a predetermined bus. By the comparison with I/O which receives other said elements and ranking numbers from a memory module through a predetermined bus, and the element which self grasps when said element and a ranking number are received A ranking number calculation means to compute the virtual ranking number which is the candidate of the ranking number of the received element, and to return a memory module besides the above, The presentation memory module of the side which is equipped with a ranking decision means to decide the ranking of an element, according to the virtual ranking number concerned, and sends out said element and a ranking number when said virtual ranking number is received, It

is attained by the information processing system characterized by deciding the ranking number of said array element by the communication link with said element and the near judgment memory module which computes a virtual ranking number by carrying out ranking number acceptance.
[0009] According to this invention, it performs through a bus with presentation of the element by the presentation memory module, and a ranking number, a virtual ranking number is computed with a judgment memory module, and the virtual ranking number concerned is given to a presentation memory module through other buses. Therefore, in a presentation memory module and a judgment memory module, it becomes possible to be able to advance sorting application in juxtaposition and to also avoid the collision of a bus.
[0010] An element specification / sending-out means to specify the element with which said memory module serves as a processing object in the desirable embodiment of this invention according to the settled ranking number, and to send out to which bus, When the same element as an element comparison means to compare the element used as the last processing object with the sent-out element is sent out It has the same value number counter which counts up the value and which shows the number of existence of the same element. When it is judged that the element with which said element comparison means became the last processing object differs from the sent-out element It is constituted so that the value of the same value number counter about the element used as the last processing object and the element concerned may be associated and any may send out. Furthermore, it has the array arranged in the sequence which received the value of the element with which which memory module became the sent-out last processing object, and a related counter, and associated and received these. According to this embodiment, in which memory module, an element and its number of duplications are received in predetermined sequence, and become possible [that this creates the array of an element without duplication, and the array which shows the number of existence of each element]. That is, the number with which each element exists in the list of elements without duplication and the array of a basis by this can be grasped easily.
[0011] According to another embodiment of this invention, said memory module said element comparison means The value number counter which counts up the value when it is judged that the element used as the last processing object differs from the sent-out element and which shows a ranking number without duplication, When the element used as the last processing object and the sent-out element are the same, about the sent-out element the value of a value number counter It was decided that it

would be the ranking number of the element concerned without duplication, on the other hand, the value of the value number counter counted up when [that] these differed was determined as the ranking number of the element concerned without duplication, and it has a renewal means of a ranking number to update the ranking number concerned. According to this embodiment, it becomes possible to change the ranking number given to the array element into the thing in the condition of having eliminated duplication of an element.
[0012] Moreover, two or more memory modules with which, as for the object of this invention, a CPU module and each have MPU and a RAM core, The connection with said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU Are the sort approach of an array of having used the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module, and it sets to the (a) memory module. The step which sorts the element which constitutes the part of the array which self grasps, (b) The location which said part which said self grasps occupies during an array is followed. The step which determines the judgment memory module of the side which receives the presentation memory module, element, and ranking number of the side which sends out an element and a ranking number among the memory modules which grasp the part of said array, (c) In a presentation memory module, the sorted element with the ranking number In the step transmitted to other memory modules through a predetermined bus, and (d) judging memory module In the step which receives other said elements and ranking numbers from a memory module through a predetermined bus, and the (e) aforementioned judging memory module Based on the ranking number of the element which the judgment memory module concerned grasps, the virtual ranking number which shows the candidate of the ranking number of the received element is computed. In the step which returns the virtual ranking number concerned to said presentation memory module, and the (f) aforementioned presentation memory module The step which updates the ranking number of an element according to the virtual ranking number concerned when said virtual ranking number is received, (g) whenever said step (d) - (f) is completed, the memory module group which consists of the presentation memory module and judgment memory module about the element to which the predetermined ranking number was given by concerned step (d) - (f) By repeating step (d) - (f) as either a presentation memory module group or a judgment memory module group, respectively

By updating the ranking number of the element in each memory module group, it is attained also by the sort approach characterized by deciding the ranking number of each element of an array.
[0013] According to the above-mentioned invention, sending out of sending out of the operation in a presentation memory module, the element in a presentation memory module, and a ranking number, the operation in a judgment memory module group, and the virtual ranking number in a presentation memory module can perform in juxtaposition, and can also avoid the collision of a bus. That is, this becomes possible remarkably to realize sorting application (ranking numbering to an array element) at a high speed. Moreover, the amount of memory to be used also becomes possible [controlling to O (N)]. In the desirable embodiment of the above-mentioned invention, a step (e) contains the step which computes a virtual ranking number based on the number of front insertion which shows the number of the elements which should be located ahead, the ranking number about the element which should be located ahead, and the received ranking number from the element received (e1). Furthermore, in a desirable embodiment, a step (f) contains the step which substitutes the virtual ranking number received (f1) for the ranking number of the element sent out at the step (c).
[0014] In the desirable embodiment of this invention, it sets in (h) presentation memory module group further. The element grasped with the memory module which constitutes the presentation memory module group concerned So that it may have the step which computes the number of duplications which shows whether it has set in the memory module group concerned, and shoes existence is recognized and it may not transmit [said step (c) overlaps the same (c1) element and] it The number of front insertion which shows the number of the elements which should be ahead located from the element with which said step (e) received the sorted element with the ranking number and the number of duplications including the step transmitted to other memory modules (e2), The step which computes a virtual ranking number is included based on the ranking number about the element which should be located ahead, and the ranking number and the number of duplications which were received. Said step (f) A virtual (f2) ranking number, Based on a difference with the ranking number at the time of sending out of the element in a step (c), the step which determines the ranking number of the same element as the element concerned is included.
[0015] According to this embodiment, a presentation memory module does not need to send out the same element repeatedly. Moreover, if the number of duplications of a

certain element is computed, the ranking number and the number of duplications of the element concerned can be transmitted to a judgment memory module, and calculation of the virtual ranking number concerning the element concerned can be performed in a judgment memory module. That is, thereby, it can prevent lowering the utilization effectiveness of a memory module.
[0016] In a still more desirable embodiment, the presentation memory module group which consists of a memory module which is 2n to which the increment of the n (n:1 or more integers) is carried out whenever a presentation memory module is an independent memory module, and a receiving module is also an independent memory module and the step of (d) - (f) ends it, and the judgment memory module group which consists of a 2n memory module are formed in first stage. If a 2n memory module is used as mentioned above, it will become suitably possible to realize sorting application.
[0017] Moreover, in another embodiment of this invention, an array is sorted by the above-mentioned sort approach. Based on the sorted array concerned, and the compile approach that the element under said array generates the new array arranged in predetermined sequence that there is no duplication (i) The step which sends out the element which serves as a processing object according to a ranking number in a predetermined memory module, (j) when the same element as the element used as the last processing object is sent out Count up the same value number counter which shows the number of existence of the same element, and on the other hand, when the element used as the last processing object and a different element are sent out The step which associates the value of the same value number counter about the element used as the last processing object, and the element concerned, and sends these out, (k) by receiving the value of the element used as the last processing object, and the same related value counter, having the step which associates these and is arranged during a new array, and repeating (l) step (i) - (j) It is characterized by associating an element and its number of existence, and being arranged during said new array.
[0018] moreover, the above-mentioned compile approach -- further -- (m) -- in which module, it has the step which acts as the monitor of the value of the element sent out at a step (j), and the same related value number counter, and a step (k) may be performed with which the module concerned.
[0019] In the memory module which grasps the array element concerned moreover, the above-mentioned compile approach -- (n) -- While forming the ranking number counter and the same value number counter which store the ranking number of the element used as a processing object, and the number of existence of the element concerned, respectively And it sets to the memory module which grasps the step

which prepares the register which stores the element used as the last processing object temporarily, and the element to which the ranking number concerned was given according to (o) ranking number. When the step which sends out the element concerned to the 1st bus, and the element received in the memory module which grasps the (p) array element are compared with the content of the register and these are in agreement While counting up the number of existence, when these are not in agreement In which memory module the step which updates the content of the register, and the value of the number counter of existence after sending out the content of the register, and the value of the number counter of existence to the 2nd bus, and (q) -- You may have the step which arranges the content of said register, and the value of the number counter of existence during an array as the number of existence of an element and the element concerned, respectively.
[0020] A step (n) contains the step which forms the value counter which stores the ranking number which does not have duplication about the element used as a processing object (n1) further. Said step (p) (p1) When the received element is compared with the content of the register and these are in agreement While giving the value of a value number counter to the ranking number of the element used as the processing object concerned When these are not in agreement, it is still more desirable to include the step which gives the value of the value number counter which counted up the value number counter and was counted up for the ranking number of the element used as a processing object.
[0021] In another embodiment of this invention, moreover, the join approach of an array of realizing share-ization of two or more arrays, using the above-mentioned sort approach and the above-mentioned compile approach (r) The step which performs processing concerning said sort approach which joins two or more arrays and gives a ranking number to each of these array elements, (s) According to the element and its ranking number under said array which joined, it has the step which generates the new array in which processing concerning said compile approach is performed and the duplicate element does not exist. That is, it becomes possible by giving the sort approach and the compile approach concerning this invention, where a desired array is merged to acquire the array which eliminated duplication of an element and by which the join was carried out.
[0022] Two or more memory modules with which a CPU module and each have MPU and a RAM core in still more nearly another embodiment, The connection with said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped

with said one or more memory modules by the instruction given to MPU of each memory module from CPU The join approach of two or more arrays of having used the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module is [0023]. In order to specify the predetermined element in the value list which is the array in which the above-mentioned sort approach and the above-mentioned compile approach are used for, and the memory module stored the element based on the record number, respectively Equip the location corresponding to a record number with the pointer array which has arranged the pointer value which shows a value list, and two or more (r1) value lists are joined. The step which performs processing concerning said sort approach which gives a ranking number to each of these array elements, (t) While generating the new value list with which processing concerning said compile approach is performed, and the duplicate element does not exist according to the element and its ranking number under said value list which joined It has the step which considers the array which consists of a ranking number of the step which updates the ranking number of said element for the ranking number of the element concerned in case the duplicate element does not exist, and an element in case the element which carried out the (u) aforementioned duplication does not exist as a new pointer array to show a new value list.
[0024]
[Embodiment of the Invention] With reference to an accompanying drawing, explanation is added per gestalt of operation of this invention below a [hardware configuration]. Drawing 1 is a block diagram which shows the configuration of the computer system concerning the gestalt of operation of this invention. As shown in drawing 1 R> 1, a computer system 10 The CPU module 12 which realizes parallel operation by single instruction, the memory module 14-1 which memorizes various data required for parallel operation, 14-2, 14-3, and --, It has the fixed memory 16 which memorizes a required program and data, the input devices 18, such as a keyboard and a mouse, the indicating equipment 20 which consists of CRT etc., and the legacy memory 22 the data of various formats etc. are remembered to be. Moreover, in a bus 24-1, 24-2, and --, transfer of the information between the circuit elements which a switch 28-1, 28-2, 28-3, --, etc. were arranged by the contact with the CPU module 12 and each memory module 14, and were chosen as it is possible. Moreover, the switch 30-1 for making connection and connection of a bus, 30-2, and -- are prepared between adjoining memory modules between the CPU module 12 and the memory module 14-1. Moreover, the switch (sign 29 reference) may be formed

between the contact of the input terminal of a memory module, and a bus, and the contact of the output terminal of the memory module concerned, and a bus. The above-mentioned switch is shown by the round mark of a broken line in drawing 1 .
[0025] Furthermore, it is desirable to prepare a single input terminal and not only a single output terminal but other one or more terminals (input/output terminal etc.) in the memory module 14. For example, in the gestalt of the 2nd operation and the gestalt of the 3rd operation which are mentioned later, processing is realized using the I/O from three or more terminals.
[0026] Between the CPU module 12 and the memory module 14, two or more buses 24-1, 24-2, 24-3, 24-4, and -- are prepared. Therefore, transfer of data etc. is possible for between the CPU module 12 and a memory module 14 and between memory modules by the above-mentioned bus. Moreover, the control signal line 25 is formed between CPU12 and a memory module 14, and the instruction emitted from CPU12 is transmitted to all the memory modules 14.
[0027] Furthermore, the local bus 26 is arranged between CPU12 and other components, and transfer of data etc. is possible for for example, fixed memory 16, an input device 18, etc. also among these. CPU12 reads the program memorized by other storage (not shown) like RAM which was memorized by fixed memory 16 or was connected on the bus 26, and performs control of the switches 28-30 besides transfer of data including sending out of the instruction to the memory module 14 shown below etc. according to this program. Moreover, CPU12 can receive the data of the various formats memorized by the legacy memory 22 according to a program, can change them into a series of data (array) which can process the data of this format by the system which consists of CPU12, a memory module 14, and a bus 24, and can also store these in each memory module 14.
[0028] Drawing 2 is a block diagram which shows the outline of each memory module 14. As shown in drawing 2 , a memory module 14 The clock buffer 32 which accepts synchronizing signals, such as a clock given from the CPU module 12, The space ID mentioned later, the element number of data, etc. are grasped as the RAM core 34 which memorizes data. MPU36 which controls data read-out from the data writing and RAM core to the RAM core 34 based on Space ID and an element number when the instruction from CPU12 etc. is received, It has I/O38 which receives the data from either of the buses, and supplies the RAM core 34, and/or sends out the data from the RAM core 34 to which bus. In the gestalt of this operation, through the control signal line 25, a memory module 14 accepts the instruction from CPU, MPU36 can answer this instruction, and can read the data of the RAM core 34, and it can write data in the

RAM core 34, or can perform predetermined processing now to data. Moreover, a data input and data output are performed based on synchronizing signals, such as a clock given to the clock buffer 32, through the data access to the RAM core 34, and I/O. As for MPU36 of the above-mentioned memory module 14, it is desirable that it consists of two or more processing units, and two or more processings can be performed in juxtaposition.
[0029] In this invention, it is possible that a computer system 10 is a system of a memory share mold so that clearly from drawing 1 and drawing 2 . Moreover, each memory module 14 performs processing in juxtaposition by giving an instruction to each memory module 14 through the control signal line 25 so that it may mention later. Moreover, the data output to a bus, the data input from a bus, etc. are performed based on a predetermined synchronizing signal. Therefore, it is possible that this computer system 10 is making the gestalt of SIMD. Fundamentally, the computer system 10 equipped with such a configuration is equipped with the multi-space memory concerning a design of this invention person indicated by Japanese Patent Application No. No. 263793 [11 to], the memory module, and the rearrangeable bus. Lessons is taken from these and explanation is simply added to below.
[0030] (1) In a multi-space memory book description, multi-space memory means the room assigned in order to access room based on Space ID and the address. Thereby, even if a series of data are alloted by many processors, each processor can separate and recognize this certainly.
[0031] In the conventional room, even if it might assign the field according to individual for every process, assigning room to every [a series of] variables (an array, structure, etc.) was not performed. Therefore, such conventional room is hereafter called "single room." In the system of single room, since data are accessed only using the address, a series of data which have relation were not able to be separated, and it has not recognized. For this reason, even if parallel processing was actually possible, that propriety was not able to be judged in many cases. Moreover, the garbage collection needed to be performed in order to secure the hold location of a series of data concerned, when making a series of new data hold in a certain single room.
[0032] On the other hand, in this invention, Space ID was introduced into room and the same ID is given to it about a series of data. Moreover, in a memory module 14, the space ID about the data currently held at the own RAM core 34 can be grasped, and, thereby, each [memory module 14 / itself] can determine the right or wrong of self actuation by referring to the space ID of the data accessed now. Moreover, since each memory module relates with Space ID and all or some of a series of data can be held,

a certain data of a series of can be divided and stored in two or more memory modules 14, and, thereby, a garbage collection can be made unnecessary.
[0033] (2) In a memory module and this invention, each memory module 14 had MPU36, and grasp each element number of a series of data which self besides the above-mentioned space ID holds. Therefore, after receiving the instruction from CPU12, the data which MPU36 should access according to an instruction can judge whether it is what is held in the RAM core 34 of self, and can determine the right or wrong of the need as access. Furthermore, each memory module 14 is able to determine the assignment range of the tacit processing in the instruction in SIMD from the range of the suffix of the array element stored in the RAM core 34 of self. The storage sequence of the element which should be processed is replaced according to the instruction from CPU12, and each memory module 14 can sort the element currently held in the RAM core 34 of self.
[0034] (3) Pipeline processing is realized by setting to rearrangeable bus this invention, and CPU's12 turning on / turning off selectively a switch 28-1, 28-2, -- and a switch 30-1, 30-2, and --, and specifying the memory module 14 which should deliver and receive data. As shown in drawing 3 , for example, the data outputted from certain memory module 14-i other memory module 14-j -- giving -- and -- being concerned -- others, when the data outputted from memory module 14-j should be transmitted to memory module 14-k of further others CPU12 sets up the condition of each switch so that bus 24-m may be assigned for memory module 14-i and 14-j and bus 24-n may be assigned for memory module 14-j and 14-k.
[0035] Furthermore, not only when connection between single memory modules realizes, but these pipeline processing can be realized by connection between two or more of a series of memory modules (memory module group). According to the processing which it is going to attain, between each memory module can be switched, and a communication link can be schedule-ized so that the capacity of a bus can be used for an one direction about 100% by carrying out a continuation transfer in the sequence that the data of the defined class were able to be defined, for every connection path. Thereby, the lowness of the performance of interprocessor communication which was the biggest problem of the parallel processing system of a distributed memory type is cancelable.
[0036] [Multi-space memory] Explanation is again added more to a detail about the memory management of each memory module in the computer system concerning this invention using multi-space memory, and the memory access according to an instruction. Drawing 4 is drawing for explaining the structure of a memory module 14

under multi-space memory. As shown in drawing 4 (a), a space ID managed table is prepared in the RAM core 34 in a memory module 14. Thereby, MPU36 of a memory module 14 becomes possible [grasping required information, such as the space ID of the data which self holds,].
[0037] As shown in drawing 4 (b), the logic starting address of a data constellation under the management of Space ID and CPU for every data constellation which self holds, the size of the field where the data constellation was assigned, the physical starting address in the RAM core 34, the total size of a series of data which have the space ID concerned, and the access-restriction flag that shows access restriction are stored in the space ID managed table. in the gestalt of this operation, reading appearance of the access-restriction flag is carried out, and a chisel is possible for it -- only (R) and writing are possible -- (R) and R/W are possible -- three conditions of (RW) can be shown now. When the data constellation which has a certain space ID is given, MPU36 of a memory module 14 finds out one or more fields which should hold the data constellation concerned into the RAM core 34, divides a data constellation into remaining as it is or 2 or more, and holds it in the field concerned. In this case, the logic starting address and allotment area size in the RAM core which held data actually with the given space ID, a logic starting address, total size, and an access-restriction flag are also memorized by the space ID managed table. Drawing 4 (c) is drawing showing the data in the RAM core 36 according to the space ID managed table by drawing 4 (b).
[0038] Explanation is added to below per [to [the outline of memory access], thus the constituted memory module 14] access. As shown in drawing 5 , CPU12 transmits a required instruction (for example, writing and read-out of data) to all the memory modules 14 through the control signal line 25 first at Space ID and the logical address, and a list. In each memory module 14, this is answered, the space comparator 52 formed in MPU36 compares Space ID with the space ID currently held on the space ID managed table of self, and it judges whether self holds the same thing, and a address comparator 54 makes the same judgment about the logical address. Subsequently, when it is judged that the data with which MPU36 of a memory module 14 serves as a processing object by the instruction at the RAM core 34 of self are held, with reference to a space ID managed table, address KARIKYURETA 56 computes the physical address in the RAM core 34, and specifies the data used as a processing object. Thus, after data are specified, MPU36 performs processing (for example, writing and read-out of data) according to the instruction given from CPU12, and when required, it transmits data to CPU12 (refer to drawing 5 (c)).

[0039] [Sorting application (gestalt of the 1st operation)] Explanation is added per [concerning the computer system 10 constituted in this way] sorting application. In addition, in the following explanation, since the memory module concerning this invention is a memory module equipped with MPU (processor), PMM (Processor Memory Module) is called.
[0040] In order to make an understanding easy, as shown in drawing 6 , four PMM(s) consider the case where two elements (family name) are held, respectively. As shown in drawing 6 (a), the family name "****" whose suffix (namely, record number) of an element is "0", and the family name "****" whose suffix is "1" are held at a certain PMM (1st PMM 14-1). the family name "******" whose suffix is "2" at 2nd PMM 14-2, and a suffix are "3" -- "-- passing -- a basis -- " -- ** -- the family name to say is held. Hereafter, the family name corresponding to a suffix as shown at drawing 6 (a) is held also at 3rd PMM 14-3 and 4th PMM 14-4, respectively. The same space ID was given to the array which consists of these elements, and each MPU36 of PMM has managed in it a suffix (record number), a physical address stored actually of the element which the RAM core 34 of self manages using the space ID managed table.
[0041] For example, the instruction which sorts the array which has this space ID through the control signal line 25 from CPU12 thinks that it was given to each PMM 14-1 to 14-4. Drawing 7 is a flow chart which shows the procedure of the sorting application concerning the gestalt of this operation. If an instruction (for example, instruction "sort the element under array which has a certain space ID") is published by CPU12 as shown in drawing 7 (step 700) This instruction is answered and it sets to each PMM. Each MPU36 of PMM The instruction given through the control signal line 25 is received. The content is interpreted (step 701), the "space ID" in an instruction is investigated (step 702), and it judges whether it relates to the space ID of the data which the RAM core 34 of self holds (step 703). When judged as a no (No) at step 703, processing is ended, and when [that] judged yes (Yes), on the other hand, whether the writing of the data constellation about the space ID concerned of MPU36 being attained with reference to the space ID managed table and a required check are performed (step 704). When it is judged by the check that it is abnormal (it is yes (Yes) at step 705), MPU36 notifies to CPU12 that the error arose through the control signal line 25. On the other hand, in [that] being normal, MPU36 performs the sorting application body described below (707 or less step).
[0042] First, each of PMM 14-1 to 14-4 relevant to processing performs the sort of the element which self holds (step 707). This sort is actually accompanied by exchange of the element in each PMM14. More specifically, MPU36 sorts the element

held in the RAM core 34 of self using the known sort technique, such as quick sort. Drawing 6 (b) is drawing showing the condition that the element under array in each PMM shown in drawing 6 (a) was sorted. In addition, as shown in drawing 6 (b), it should care about that arrangement of the suffix (record number) of each element is also changed with the sort of the above-mentioned element.
[0043] subsequently, every -- only the number of the elements under array which self has held / managed secures the field (ranking number field) for arranging a ranking number, and, as for MPU36 of PMM14, gives the initial value of each ranking number (step 708). Drawing 6 (c) is drawing showing each condition that the initial value of a ranking number was given about PMM. Thus, a ranking number is given in first stage within the element sorted within each module. Subsequently, merge between adjoining pairs and ranking numbering are performed (step 709). In step 709, first, CPU12 controls the switches 28 and 30 on a bus 24, and connects one output and the input of another side to one input of a predetermined pair, the output of another side, and a list among PMM(s) relevant to sorting application. Also when [adjoining / two] it does not PMM and adjoin, as for the above-mentioned pair, it is desirable to consist of two PMM(s) located in near. For example, in drawing 1 , when the thing relevant to sorting application is PMM 14-1 to 14-4, it is desirable to make PMM 14-3 and 14-4 into a pair by making PMM 14-1 and 14-2 into a pair. As shown in drawing 8 , CPU12 so that the output of PMM 14-1 and PMM 14-2 may be connected to a bus 24-1 And so that the input of PMM 14-1 and the output of PMM 14-2 may be connected to a bus 24-2 A switch 28 is controlled to connect the input of PMM 14-3, and the output of PMM 14-4 to a bus 24-2 to control a switch 28 and to connect the output of PMM 14-3, and PMM 14-4 to a bus 24-1. Furthermore, CPU12 turns OFF further the bus 24-1 arranged between PMM 14-2 and PMM 14-3, the switch 30-5 on 24-2, and 30-6. In drawing 8 , the condition that what is expressed with the black dot has flowed is shown, and the condition that what is expressed with a circle [white] is not connected with a flow thru/or PMM is shown. Moreover, other things follow the condition of other PMM(s) (not shown). In addition, in the example of drawing 8 , having divided by turning a bus 24-1 and 24-2 a switch 30-5, and turning OFF 30-6, and using the bus for validity more could understand.
[0044] Thus, if connection between PMM(s) is prescribed by CPU12 as typically shown in drawing 9 , the processing body of ranking numbering between the pairs of PMM will be performed. Drawing 10 thru/or drawing 12 are drawings showing typically ranking numbering about the array shown by drawing 6 , in order to make an understanding easy, and it is a flow chart which shows the ranking number processing

between the pairs of PMM with more common drawing 13 . In drawing 10 R> 0 thru/or drawing 12 , although only the processing process in PMM 14-1 and PMM 14-2 was shown, processing in PMM 14-3 and PMM 14-4 is also performed in juxtaposition. In addition, in processing, what gives data first to PMM of another side is called PMM of the first half, and what is received (PMM of another side) is called PMM of the second half here. Since PMM of the first half presents an element and a ranking number, it can be called presentation PMM, and since it judges the ranking number shown PMM of the second half on the other hand, it can be called judgment PMM. Which PMM may turn into PMM of the first half among pairs. In this example, for convenience, PMM 14-1 turns into PMM of the first half, and PMM 14-2 is PMM of the second half.
[0045] First, in PMM of the first half, the pointer (a "PUT pointer" is called hereafter) in which a processing location is shown is arranged to an initial position (it sets into the part of the sorted array and is a head, i.e., the location of the "0th" watch). On the other hand, the element received from PMM of the first half and the pointer (a "comparison pointer" is called hereafter) in which the location which should be compared first is shown are arranged to an initial position (it sets into the part of the sorted array and is a head, i.e., the location of the "0th" watch) so that it may explain below in PMM of the second half (drawing 10 (a) and step 1301 of drawing 13 , 1311 reference). In the gestalt of this operation, the comparison pointer used in PMM of the second half has taken the gestalt of the array of structures (X, Y, Z). X shows the head location (that is, an "unsettled location" is called the head location of a non-compared element, and henceforth) which should be compared here. Y The total of the element received from PMM of the first half is shown ("the number of front insertion" is hereafter called by the case.). Z is the proposal (a "virtual ranking number" is hereafter called by the case.) of the ranking number of the element given from PMM of the first half in the imagination array which merged PMM of the first half, and PMM of the second half, and was acquired. It is shown.
[0046] Subsequently, the first data transfer is performed by MPU of PMM of the first half. In this data transfer, the element of the location which a PUT pointer shows is transmitted to PMM of the second half through a bus (drawing 10 (b) and step 1303, 1312 reference). In addition, in branching of step 1302, although always judged yes (Yes) in processing between two PMM(s), about this, it mentions later. In the first data transfer, an element "****" is transmitted to PMM of the second half. In PMM of the second half, the location which should insert the transmitted element "****" is discovered in the part of the array stored in PMM of the second half (step 1313). Actually, this should just discover the location which should be inserted rather than

inserts a value. In the gestalt of this operation, the element stored in the part of each array of PMM is arranged in the condition of having sorted actually. Therefore, retrieval of an insertion point is realizable using the high-speed search technique, such as the BAISE cushion method (split half method). By discovering an insertion point, it is the element which ranking has not decided and it becomes possible to pinpoint the range of the element ahead located from an insertion point (for "the range 1" to be called hereafter). In addition, in the gestalt of this operation, when there is the same element, the agreement that the ranking of PMM of the first half has priority is carried out. Therefore, when the element "****" transmitted from PMM of the first half exists also in PMM of the second half, the ranking of the direction stored in PMM in the first half is that priority is given (that is, smaller ranking number).
[0047] In this example, it turns out that the element "****" transmitted from PMM of the first half is located ahead of an element "****" among the part of the array which PMM of the second half grasps, and it turns out that the element belonging to the range "1" does not exist by this (it sets to step 1314 and is [refer to drawing 10 (c) and] yes (Yes)). Then, MPU of PMM of the second half returns "0 (namely, head)" to PMM of the first half through the bus of another side as a ranking number of the transmitted element "****" (step 1315). Subsequently, MPU of PMM of the second half increments a virtual ranking number, and sets it to "1" while it increments the number of front insertion and sets it to "1" (step 1316). It is because it is necessary to increment the number of front insertion and since one element transmitted from front PMM increased this, and the ranking number of the following element needs to increment what was given at least this time (in this case, "0"). If the ranking number (insertion point) of an element is given from PMM of the second half (step 1332), MPU of PMM of the first half will store the given ranking number as a ranking number of the corresponding element (step 1334), and, subsequently will increment a PUT pointer (drawing 11 (a) and step 1335 reference). Thus, the ranking of a certain element in PMM of the first half is decided.
[0048] Next, MPU of PMM of the first half transmits the element "****" of the location which a PUT pointer shows to PMM of the second half through a bus (drawing 11 (b) and step 1303 reference). In PMM of the second half, the location which should insert the transmitted element "****" is previously discovered like the time of an element "****" being transmitted (step 1313). It turns out that an element "****" is located behind an element "** et al." among the part of the array which PMM of the second half grasps (it is yes (Yes) at drawing 11 (c) and step 1314). Thereby, in the part of the array which PMM of the second half grasps, the ranking of

each element can be decided in the number of the elements located an element "** et al." and ahead [its], and a list. MPU of PMM of the second half makes a detail decide the ranking of the above-mentioned element in the following procedures more.
[0049] First, the number of front insertion "Y" is applied to the ranking number about the element contained in the range "1", respectively (step 1317). Thereby, the ranking of the element contained in the range "1" is decided. In the example mentioned above, "0+1=1" and the ranking number of an element "** et al." are set to "1+1=2" by the ranking number of an element "****." Subsequently, an unsettled location is changed into the location of the next element of the element at the tail end in the range 1 while the ranking number of the element at the tail end is substituted for a virtual number among the elements contained in the range 1 (step 1318) (step 1319). In the above-mentioned example, the ranking number "2" of an element "** et al." is given to Z of a comparison pointer (array of structures), and an unsettled location is changed into "2" from "0." Thereby, an array of structures is set to (2, 1, 2). The increment of the number of front insertion "Y" and virtual ranking number "Z" after such processing and in an array of structures is carried out (step 1320). Thereby, an array of structures is set to (2, 2, 3) (refer to drawing 11 (d)). The virtual ranking number obtained at step 1320 turns into a ranking number of the element (at the above-mentioned example, it is also "****") received at step 1312, and MPU of PMM of the second half transmits the ranking number (the above-mentioned example "3") concerned to PMM of the first half (step 1321). After such processing, the increment of the virtual ranking number is carried out further (step 1322). This is because the ranking number of the following element will become bigger [one] at least than the ranking number given this time.
[0050] PMM of the first half stores the received ranking number as a ranking number of the corresponding element, and, subsequently increments a PUT pointer. Thus, the ranking number of the element in PMM of the first half is decided. In PMM of the first half, the value MPU of PMM of the first half indicates termination to be for an unsettled element not to already exist (that is, a ranking number is decided about all elements and the element is not arranged in the location of a PUT pointer) is transmitted to PMM of the second half (step 1306 reference). The value which shows termination here is a bigger value than the value which shows the element at the tail end of an array. PMM of the second half answers acceptance of the value which shows the above-mentioned termination, and performs processing of the same processing (steps 1312-1322 of drawing 13) as abbreviation. In the above-mentioned example, since the element contained in the range "1" in spite of acceptance of the

value which shows termination does not exist, step 1323 is reached through steps 1315 and 1316, and processing is ended (refer to drawing 12 (b)).
[0051] In PMM of the first half, processing is completed by sending out (step 1316 reference) of the value which shows termination, and decision (it is yes (Yes) at step 1336) of the ranking number of all elements. In the same procedure as the above-mentioned processing, a merge application is performed also between PMM 14-3 and PMM 14-4, and thereby, as shown in drawing 12 (c), the ranking number of each element is decided.
[0052] If the sequence number of each element in two PMM(s) is decided, CPU12 will switch a switch and will connect between two PMM groups which each becomes from two PMM(s). Drawing 14 (a) and drawing 14 (b) are drawings showing an example of connection of two PMM groups in PMM shown in drawing 8 , respectively. In drawing 14 (a), PMM 14-1 and 14-2 constitute the 1st PMM group. CPU12 PMM 14-3 and 14-4 constitute the 2nd PMM group. PMM 14-1 and the output of 14-2, Switches 28 and 30 are controlled so that the input of the PMM group 14-3 is connected, and the output of PMM 14-3 and the input of PMM 14-4 are connected and the output of PMM 14-4, and PMM 14-1 and the input of 14-2 are connected (step 709 reference of drawing 7). Or to be shown in drawing 14 (b), a switch may be controlled so that PMM 14-1 and the output of 14-2 are connected with PMM 14-3 and 14-4.
[0053] Drawing 15 (a) and (b) are drawings which expressed typically drawing 14 R> 4 (a) and (b), respectively. Although it becomes clear behind Data given to PMM 14-1 from PMM 14-4, and 14-2 in drawing 15 (a) (among drawing) Referring to the sign ** is data (among drawing) which show a ranking number and are given to PMM 14-3 from PMM 14-1 and 14-2. The data (refer to sign ** among drawing) which referring to the sign ** shows an element, and are given to PMM 14-4 from PMM 14-3 show the virtual ranking number which an element and PMM 14-3 computed. moreover, drawing 15 (b) -- also setting -- every -- data ** delivered and received between PMM(s) and ** are the same as the thing of drawing 15 (a), and, on the other hand, the data (refer to sign **) transmitted to PMM 14-4 from PMM 14-3 shows the virtual ranking number which PMM 14-3 computed.
[0054] As shown in above-mentioned drawing 12 , explanation is added about the processing (step 709 reference of drawing 7) which determines the merge application in two PMM groups, and the ranking number of an array from the ranking number of the element contained in the part of an array and these in the pair of two PMM(s). in addition, the connection voice of the bus shown in drawing 14 (b) and drawing 15 (b) by the following explanation -- like -- following -- every -- the processing performed

in PMM is explained.
[0055] First, each arranges a PUT pointer to an initial position in PMM 14-1 and PMM 14-2 ("the PMM group of the first half" is called hereafter.) (step 1301). In addition, in future processings, a PUT pointer moves in PMM which constitutes the PPM group of the first half according to the element which self holds being sent out. Each of the PMM which it is in the second half on the other hand arranges a comparison pointer to an initial position while initializing the array of structures (step 1302). Subsequently, in each PPM which constitutes the PMM group of the first half, each PMM which constitutes current and the PMM group of the first half grasps of which ranking number the element was sent out. In addition, fundamentally, although the transmitting pointer, receiving pointer, and both sides which use at the time of transmission are used as a PUT pointer in the flow chart, migration of these transmitting pointer and a receiving pointer is performed only with few time difference, although the processing time in the PMM group of the second half is inserted. For example, when a transmitting pointer is made into an increment in a certain PMM so that it may mention later (step 1304 reference), the PMM concerned increments a receiving pointer also in reception (step 1335 reference).
[0056] Each PMM which constitutes the PPM group of the first half judges whether the element concerned is what self is holding based on the ranking number of the element set as the object of processing (step 1302). When judged yes (Yes) at this step 1302, the element to which a PUT pointer points is transmitted to PMM 14-3 and 14-4 through a bus 24 (refer to step 1302 of drawing 13 , and drawing 16 (a)). In the above-mentioned example, the element "****" which is a ranking number "0" is first transmitted to PMM 14-3 from PMM 14-1, and PMM 14-4. By this processing, the location of a PUT pointer moves in PMM 14-1 (step 1304).
[0057] Respectively PMM 14-3 and PMM 14-4 receive an element (step 1312), discover the location which should insert the element (step 1313), and judge whether the element belonging to the range "1" exists (step 1314). About the above-mentioned element "****", it is judged as a no (No) in step 1314. Thereby, in PMM 14-3, since the virtual ranking number of an element "****" is set to "0", this value is transmitted to PMM 14-4. The virtual ranking number of an element "****" is set to "0" also in PMM 14-4. Then, MPU of PMM 14-4 returns "MAX(0 0) =0" to a front PMM group through a bus as a ranking number of an element "****" (drawing 16 (b) and step 1315 reference). Subsequently, in PMM 14-3 and 14-4, the number of front insertion (Y) and virtual ranking number (Z) in an array of structures increment, respectively (step 1316). Thereby in the above-mentioned example, each array of

structures is set to (0, 1, 1), and (0, 1, 1).
[0058] If a ranking number is given from the PMM group of the second half (step 1331), it will judge whether each PMM which constitutes the PMM group of the first half is what self holds [the element under current processing (for example, element ″****″)] (step 1333). When the ranking number of an element ″****″ is transmitted, PMM 14-1 judges yes (Yes) at the above-mentioned step 1333, and rewrites the ranking number corresponding to the element of the location to what it was given from the PMM group of the second half (drawing 16 (a) and step 1334 reference). Similarly, the PMM group of the first half transmits the element to which the following ranking number was given to the PMM group of the second half. In the above-mentioned example, from PMM 14-2, an element ″****″ is transmitted (refer to drawing 17 (a)), and a big thing ″MAX(1 1) =1″ is transmitted to the PMM group of the first half as a ranking number of the element ″****″ concerned among each virtual ranking numbers of the PMM group of the second half (refer to drawing 17 R> 7 (b)). Moreover, in PMM 14-3 which constitutes the PMM group of the second half, and 14-4, an array of structures is set to (0, 2, 2), and (0, 2, 2), respectively (refer to drawing 17 (b)).
[0059] Furthermore, the PMM group of the first half transmits the element to which the following ranking number was given to PMM of the second half. In the above-mentioned example, an element ″** et al.″ is transmitted from PMM 14-2 (refer to drawing 18 (a)). In PMM 14-3, an element ″** et al.″ is judged to be back from the element ″a shelf″ which PMM 14-3 concerned holds (step 1313 reference). therefore, PMM 14-3 -- setting -- the range ″1″ -- an element -- ″-- ** -- obtaining -- ″ -- since [and] an element ″is it a shelf?″ belongs -- an element -- ″-- ** -- obtaining -- ″ -- and the number of front insertion ″Y (= 2)″ is applied to the ranking number of an element ″is it a shelf?″, respectively. thereby -- an element -- ″-- ** -- obtaining -- ″ -- ″0+2=2″ and the ranking number of an element ″is it a shelf?″ are determined for a ranking number as ″2+2=4″ (step 1317 reference). Subsequently, MPU of PMM 14-3 gives the ranking number ″4″ of the element of the tail in the range ″1″ to the virtual ranking number Z of an array of structures (the current value is (0, 2, 2)) (step 1318 reference), and advances an unsettled location (step 1319 reference). (that is, the value of X is set to ″2″ from ″0″) Furthermore, MPU of PMM 14-3 is an array of structures (the current value increments the number of front insertion ″Y″ and virtual ranking number ″Z″ of (2, 2, 4) (step 1321 reference).). Thereby, an array of structures is set to (2, 3, 5). The virtual ranking number ″Z (= 5)″ in PMM 14-3 is transmitted to PMM 14-4 through a bus. After that, MPU of PMM 14-3 increments the virtual ranking number ″Z″ of an array of

structures (step 1322 reference). In the above-mentioned example, an array of structures is set to (2, 3, 6) by giving step 1322.
[0060] the element with which PMM 14-4 concerned, on the other hand, holds an element "** et al." in PMM 14-4 -- "-- passing -- a basis -- " -- it is judged that it is located while "being means" (step 1313 reference). therefore, PMM 14-4 -- setting -- the range "1" -- an element -- "-- passing -- a basis -- " -- since it belongs -- an element -- "-- passing -- a basis -- " -- the number of front insertion "Y (= 2)" adds to a ranking number -- having -- thereby -- an element -- "-- passing -- a basis -- " -- a ranking number is determined as "1+2=3" (step 1317 reference). Subsequently, MPU of PMM 14-4 gives the ranking number "3" of the element of the tail in the range "1" to the virtual ranking number "Z" of an array of structures (the current value is (0, 2, 2)) (step 1318 reference), and advances an unsettled location to it (step 1319 reference). (that is, the value of "X" is set to "1" from "0") Furthermore, MPU of PMM 14-4 increments the number of front insertion "Y" and virtual ranking number "Z" of an array of structures (the current value is (1, 2, 3)) (step 1321 reference). Thereby, an array of structures is set to (1, 3, 4).
[0061] The virtual ranking number to which PMM 14-4 was given from PMM 14-3 next "Z (= 5)", The virtual ranking number "Z (= 4)" which self computed is compared, and "MAX(5 4) =5" which is the value of the bigger one is transmitted to the PMM group of the first half as ranking of the element "** et al." to which it was transmitted (step 321 reference). Thereby, in the PMM group of the first half (setting to PMM 12-2 which sent out the element "** et al." to the detail), it is decided that the ranking number of the element concerned is "5." In addition, also in PMM 14-4, the increment of the virtual ranking number in an array of structures "Z" is carried out after step 1321 (step 1322 reference). An array of structures is set to (1, 3, 5) in the above-mentioned example.
[0062] similarly, an element "****" sends out from the PMM group of the first half -- having (drawing 19 (a)) -- processing in this case is also performed according to drawing 13 . If it explains briefly again, since the element which belongs ahead in the range "1" from the insertion point of an element "****" does not exist in PMM 14-3 which received the element "****", PMM 14-3 transmits the virtual ranking number in the array of structures "Z (= 6)" to PMM-4. Since the element which belongs ahead in the range "1" from the insertion point of an element "****" does not exist in PMM 14-4, The virtual ranking number in the array of structures "Z (= 5)" is compared with the transmitted virtual ranking number "Z (= 6)", and the bigger one (MAX(6 5) =6) of it is returned to the PMM group of the first half as a ranking number of an element

"****" (drawing 19 (b) and step 1315 reference). In the PMM group of the first half, PMM 14-1 which sent out the element "****" rewrites the ranking number corresponding to an element "****" for the received ranking number (= 6). In addition, in PMM 14-3, by passing through step 1316, the array of structures is set to (2, 4, 7), and, on the other hand, the array of structures is set to (1, 4, 6) by [the] passing through step 1316 in PMM 14-4.
[0063] Thus, after sending out of all elements is completed in the PMM group of the first half, which PMM which constitutes the PMM group of the first half transmits the value which shows termination to the PMM group of the second half (step 1306 reference). Each PMM which constitutes the PMM group of the second half receives this, and performs processing of step 1312 thru/or step 1323, respectively. In the above-mentioned example, the element "they are means" which has not decided ranking exists in PMM 13-4. For this reason, in PMM 13-4, in step 1314, it is judged yes (Yes), and the number of front insertion "Y" is applied to the ranking number of the element "they are means" belonging to the range "1", and let "3+4=7" and the obtained number "7" be the ranking numbers of an element "they are means." After passing through such processing, in each PMM which constitutes the PMM group of the second half, it is judged yes (Yes) at step 1323, and the processing in the PMM group of the second half is also ended.
[0064] the case where the element under array is stored in PMM beyond it although the element under array was stored in four PMM(s) in the above-mentioned example -- further -- four PMM(s) -- a group, as PMM, each creates the pair of the PMM group which consists of four PMM(s), and should just perform the same processing as abbreviation among these pairs. For example, as shown in drawing 20 , I think that the element under a certain array is stored in 1024 PMM(s). PMM1, PMM2 and PMM3 and PMM4, PMM5 and PMM6, --PMM1023, and PMM1024 are connected first, respectively (refer to the continuous line between PMM(s)). In this case, between these two PMM(s) The ranking number of an element is decided. Subsequently PMM1 and PMM2 The PMM group of the first half, The pair of the PMM group which makes PMM3 and PMM4 the PMM group of the second half, and PMM5 and PMM6 The PMM group of the first half, The pair of the PMM group which makes PMM7 and PMM8 (not shown) the PMM group of the second half, -- The pair of the PMM group which makes PMM1021 and PMM1022 (not shown) the PMM group of the first half, and makes PMM1023 and PMM1024 the PMM group of the second half is formed. Between each pair is connected (refer to broken line), and the ranking number of an element is decided between two PMM groups which constitute these pairs. The pair of the PMM

group which makes hereafter four PMM(s) which follow the PMM group of the first half, and this in four PMM(s) the PMM group of the second half (refer to alternate long and short dash line), Like the pair (refer to dotted line) of the PMM group which makes eight PMM(s) which follow the PMM group of the first half, and this in eight PMM(s) the PMM group of the second half, each carries out sequential formation of the pair of the PMM group which consists of a PMM group which is 2n, and decides the ranking number of an element among these. It becomes possible by deciding the ranking number of an element to decide the ranking numbers of all the elements in 1024 PMM(s) between the pairs of the PMM group which makes eventually 512 PMM(s) which follow the PMM group of the first half, and it in 512 PMM(s) the PMM group of the second half.
[0065] Thus, the pair of the PMM group which each becomes from 2nPMM is formed. By carrying out sequential decision of the ranking number of the element stored in each PMM of the PMM group which constitutes a pair, (Step 709 of drawing 7 , 710 reference), If the ranking number of all elements is decided eventually (yes (Yes), processing which carries out the reconstititution of the array according to the above-mentioned ranking numbering is performed at step 710 when required (step 711).) Although this processing is not indispensable, it becomes possible to realize more information processing performed behind at a high speed by generating an array by which the element is arranged according to the ranking number.
[0066] In a detail, CPU12 controls switches 28 and 30 first more to connect with the bus by which the input and output of each PMM are. Drawing 21 is drawing showing connection between these typically, when PMM is four. Subsequently, MPU of PMM 14-1 to 14-4 emits an element and a ranking number on a bus according to the settled ranking number. Each MPU acts as the monitor of the element emitted on a bus, and its ranking number, incorporates the element which has the same ranking number as the suffix (record number) of the element alloted with the RAM core of self from the first, and stores it in the predetermined field of a RAM core. For example, what is necessary is to incorporate the element to which a ranking number "0" and "1" were given, and just to memorize these in PMM which had memorized the suffix (record number) "0" and the element of "1" to the RAM core of self from the first (for example, 14 to PMM1 reference of drawing 10). If it does in this way, it will become possible to allot each array actually sorted in PMM. In addition, also in case the sorted array is formed in this way, MPU of PMM creates a required space ID managed table.
[0067] Or other PMM(s) (PMM14-5-PMM 14-8) for alloting the sorted array, as shown in drawing 22 are prepared. From PMM14-1-PMM 14-4, each of other PMM groups

may act as the monitor of the element by which a sequential output is carried out, and its ranking number, may incorporate the element which self should incorporate according to a ranking number, and may memorize to each RAM core of PMM. For example, 1024 PMM(s) are prepared using above-mentioned this invention, and when [each] about 1 million data (element) are stored in PMM and these data are sorted, it is thought that a sort is completed by the following time amount. All PMM(s) that as for the bus which connects between each PMM all PMM(s) operate here in juxtaposition (that is, PMM which is not performing processing does not exist), and relate to it during processing possible [6.4GB/second of data transmission] assume that it can said-operate that it is simultaneous and cooperatively. Moreover, I think that the sort of each about 1 million data (element) in PMM is completed in 2.5 seconds. In this case, in order to sort about 1 billion elements in 1024 PMM(s), it turns out that it needs only about abbreviation 4 second.
[0068] According to the gestalt of this operation, each PMM is divided into the pair of two PMM(s) in first stage, subsequently, it divides into the pair of the PMM group by which each group is constituted from 2nPMM, and the ranking number is made to decide between each pair one by one. Moreover, decision of the ranking number in each pair can perform in juxtaposition by adjusting the bus used in each pair using a switch etc. Furthermore, the ranking number in each pair can be decided by repeating the procedure of transmitting the ranking which transmitted the element from the PMM group of the first half to the PMM group of the second half, was made deciding it according to the value in the array of structures in the PMM group of the second half, and was decided to the PMM group of the first half. Therefore, while being able to perform processing very in juxtaposition, without PMM ("it playing") which is not performing processing arising, the amount of data transfer using a bus is reducible. [so-called] This becomes possible to make a sort rate into a high speed remarkably.
[0069] In addition, in the gestalt of implementation of the above 1st, as shown in drawing 14 (b) and drawing 15 (b), PMM is connected, and although sorting application is realized to ****** which gives a ranking number to each element among these, as shown in drawing 14 (a) and drawing 15 R> 5 (a), PMM may be connected. In this case, if processing (step 1312 - step 1323) of the PMM group of the second half in drawing 13 is not performed in juxtaposition but a virtual sequential number is obtained in a certain PMM, the element used as a processing object and the virtual ranking number concerned will be transmitted to adjoining PMM, and processing of steps 1312-1323 will be performed in the PMM concerned. Therefore, if the number of PMM which constitutes the PMM group of the second half increases, delay of processing may be

caused so much.
[0070] Explanation is added per gestalt of sorting application (gestalt of the 2nd operation)] besides [, next operation of the 2nd of this invention. In the gestalt of implementation of the above 1st, all elements (element in the PMM group of the first half) are transmitted to the PMM group of the second half. However, many duplication values may appear as an array becomes huge. By the technique concerning the gestalt of implementation of the above 1st, the element which takes the same value is repeatedly sent out on a bus. It is possible that it is useless to repeat and send out the element of the same value depending on the case. Then, in the gestalt of the 2nd operation, it has prevented repeating the overlapping element and sending out on a bus by counting the number of the element in a PMM group beforehand, and sending out the number to the PMM group of the second half with an element.
[0071] For example, it considers sorting application being completed in a pair of each of four PMM(s), connecting these pairs, and performing sorting application in eight PMM(s). In this case, as shown in drawing 23 , it is desirable that transfer of the data between PMM(s) can be performed using other buses (the bus 2304, 2305 reference which are located in a PMM upside in drawing 23) besides [which performs merge and sorting application of eight PMM(s)] a bus (the bus located in the PMM bottom in drawing 23 , for example, 2301 to sign 2303 reference). In a connection mode as shown in drawing 2323 , explanation is added per [which computes the number of duplications of the value in PMM14-1-PMM 14-4 (PPM / "PPM 14-1" / thru/or "PMM 14-4" are hereafter called for convenience PMM / "PMM1" / thru/or "PMM4", respectively.)] processing. The bus (sign 2304 reference) connected with the input/output terminal (I/O) of PMM1-PMM4 is called the 1st bus here, and the bus (sign 2305 reference) connected with other input/output terminals (I/O) of PM1-PMM4 is called the 2nd bus. The 1st bus is used for information interchange of the PMM group which consists of PMM1-PMM4, and the 2nd bus is used in order to give a value and its number of duplications to each PMM.
[0072] In addition, in the following explanation, as shown in drawing 25 , although the ranking number was given to each element, the number of duplications is computed in the array in PMM1-PMM4. That is, if it computes only in the PMM group of the first half, it is sufficient for the number of duplications. Drawing 24 is a flow chart which shows the processing for computing the number of duplications in a PMM group. Each of PMM1-PMM4 performs processing of various initialization first (step 2401). The ranking number counter which shows the ranking number of each value (element) applied to processing in PMM here, The same value number counter with which a

certain value (element) shows whether which overlaps and it exists, And a value-preserving register is prepared last time holding the value (element) which became a processing object in the last processing, and initial value ″0″ is given to the value of a ranking number counter and the same value number counter (refer to drawing 25). In addition, no values are held last time in first stage at a value-preserving register.
[0073] subsequently, every -- PMM specifies the ranking number of the element used as a processing object with reference to a ranking number counter, and judges whether the element to which the ranking number concerned was given is what self holds (step 2403). In the above-mentioned example, in first stage, since the counter value of a ranking number counter is ″0″, PMM3 judges that the element which self holds is a processing object (being step 2403 yes (Yes)). In addition, the following steps 2404-2405 are disregarded by the first processing (namely, processing about the element of a ranking number ″0″). The number of the existence in self-PMM the element as the element (in this case, ″****″) to which the ranking number ″0″ was given with same PMM3 indicates it to be how many it has an element ″****″ and this element into the 1st bus by judging how many it exists in (that is, how many is PMM3 holding the element ″****″?) is sent out (step 2406). In other PMM(s) (PMM1, PMM2, and PMM4), since it is judged as a no (No) at step 2403, it progresses to step 2407.
[0074] Each PMM receives the data given through the 1st bus, and applies the number of the existence in PMM to the counter value of a ranking number counter based on the number of the existence in self-PMM in data (step 2408). The counter value of a ranking number counter is set to ″0+1=1″ in the above-mentioned example. Subsequently, it is judged whether the given element differs from the thing of a value-preserving register last time (step 2409), and when both sides are the same, the number of the existence in self-PMM is applied to the counter value of the same value number counter (step 2410), and when it is the new value, on the other hand, exchange processing mentioned later is performed (step 2411). In addition, in first-time processing, since the value is not held at all last time at a value-preserving register, while decision of the above-mentioned step 2409 is omitted and an element is held in a value-preserving register last time, count-up of the same value number counter is performed. Therefore, in the above-mentioned example, each PMM sets the same value number counter to ″0+1=1″ while memorizing the received element ″****″ to a value-preserving register last time (refer to the drawing 2626). After processing of such steps 2401-2411 is repeated and the processing about the last element is completed, in step 2401, it is judged yes (Yes), and progresses to step 2412.

[0075] if processing of the first steps 2401-2411 is completed in the above-mentioned example -- every -- PMM checks that a counter value is "1" with reference to the counter value of a ranking number counter. Thereby, it turns out that PMM4 is holding the element of a ranking number "1." Moreover, since PMM4 compares last time the element "****" to which the value (element "****") and ranking number "1" of a value-preserving register were given (step 2404) and there is no change in a value, an element "****" and the number of the existence in self-PMM "1" are sent out to the 1st bus (step 2405). Since the value with which each PMM which received data through the 1st bus counted up the ranking number counter (1+1=2) (step 2408), and was remembered to be by the value-preserving register last time as shown in drawing 27 , and the received element are the same, the same value number counter is counted up (step 2410). (1+1=2)
[0076] In each PMM, processing about the element of a ranking number "2" is performed by next. By processing of the element of a ranking number "2", since PMM1 holds the element, PMM1 compares an element "******" with the element "****" memorized by the value-preserving register last time. Here, since a value has change (it is yes (Yes) at step 2404), PMM1 sends out the content (element "****") of the value-preserving register, and the value "2" of the same value number counter to the 2nd bus last time (step 2405). The content and counter value of this register are given to each PMM. When the number of duplications of a certain element (in this case, element "****") is computed so that it may mention later, sorting application (refer to drawing 31) about the element concerned may be performed. Therefore, what is necessary is just to hold an element and its number of duplications in each PMM, until the sorting application about the element concerned is completed. Moreover, an element "******" and the number of the existence in self-PMM "1" are given to the 1st bus (step 2406).
[0077] Each PMM counts up a ranking number counter based on the data given through the 1st bus (step 2408). (2+1=3) since [moreover,] the elements "******" delivered the value "****" of a value-preserving register differ last time (it is yes (Yes) at step 2409) -- every -- PMM -- last time -- the value of a value-preserving register -- rewriting (it updating) -- it transposes to the number of the existence in self-PMM to which the value of the same value number counter was given through the 1st bus (refer to step 2411 and drawing 28 (a)).
[0078] Same processing is performed also with the element of other ranking numbers. For example, about a ranking number "3", PMM3 sends out an element "******" to the 1st bus according to steps 2404 and 2406, and each PMM counts up each counter

according to steps 2407, 2408, 2409, and 2410 (refer to drawing 28 (b)). Moreover, while PMM1 sends out the counter value "2" of an element "******" and the same value number counter to the 2nd bus about a ranking number "4" according to steps 2404, 2405, and 2406 an element -- "-- passing -- a basis -- " -- the 1st bus -- sending out -- and every -- in order of steps 2407, 2408, 2409, and 2411, PMM updates a register, while counting up each counter (refer to drawing 29 (a)).
[0079] the element with which self holds PMM2 in the processing about a ranking number "5" -- "-- passing -- a basis -- " -- the 1st bus since there are two -- an element -- "-- passing -- a basis -- " -- and the number of the existence in self-PMM "2" is sent out. Therefore, in each PMM, "2" is added to each counter value of a ranking number counter and the same value number counter (refer to drawing 29 R> 9 (b)). Moreover, by this processing, since the counter value of a ranking number counter changes to "7" from "5", please care about that the ranking number of the element used as the following processing object is set to "7" instead of "6." After the processing (refer to drawing 30 (a)) about the last element to which the ranking number "7" was given is completed, it is judged yes (Yes) at step 2401. then, top PMM (the above-mentioned example PMM1) -- the 2nd bus -- an element -- "-- passing -- a basis -- " -- and the counter value "4" of the same value number counter is sent out (step 2413), and the data in which it is shown subsequently to the 2nd bus that processing was completed are sent out (step 2414). The number of existence which shows each element and its number is given to each PMM through the 2nd bus, and this is used for sorting application. In addition, what is necessary is not to be limited to this and just to define PMM which outputs beforehand data in which termination is shown, such as the last element, although top PMM consisted of above-mentioned examples so that steps 2413 and 2414 might be performed. As mentioned above, in case a PMM group is merged with other PMM groups and these elements are sorted by obtaining the number of existence of each element in a certain PMM group, it becomes unnecessary to send the duplicate element.
[0080] Drawing 31 is a flow chart which shows the sorting application which eliminated sending out of the duplicate element. Drawing 31 is the same as that of processing of drawing 13 except for a part, and the thing with the double figures same tail serves as processing which carries out an abbreviation response. Moreover, in drawing 31 , it is shown that the processing which attached the enclosure of a duplex is the processing added newly or different processing that to which drawing 13 corresponds, and a little. In this processing, PMM which holds the element (namely, element directed by the sending-out pointer) used as a processing object sends out the number of

duplications of the element concerned in the PMM group of the first half (the number of existence) "N" to the PMM group of the second half with that element in the PMM group of the first half (step 3103, 3103-2 reference). For example, in the example shown in drawing 25 thru/or drawing 30 , from the PMM group of the first half which consists of PMM1-PMM4, when an element "****" is sent out to the PMM group of the second half, the number of duplications "2" of the element "****" in the PMM group of the first half besides an element "****" is transmitted. Moreover, in sending-out processing of the PMM group of the first half, only the number of the elements concerned which self grasps moves [PMM / which outputted an element and its number of duplications] a sending-out pointer after the output (step 3104). For example, as shown in drawing 28 (a), the number of duplications of an element "****" is "2", and these [one / every] are grasped in PMM3 and PMM4. Therefore, in PMM3 and PMM4, one location of a sending-out pointer moves caudad, respectively. in addition, every -- total of the movement magnitude of the sending-out pointer in PMM becomes equal to the number of duplications of the element concerned "N."
[0081] In each PMM which constitutes the PMM group of the second half which, on the other hand, received an element and its number of duplications, as shown in step 3116 and step 3120 of drawing 31 R> 1, the number of duplications "N" is applied to the number of front insertion, and a virtual ranking number, respectively. This supports that the element (ranking is small) ahead located from itself exists only in "N."
[0082] Furthermore, in the reception of PMM which constitutes the PMM group of the first half, the difference "M" of the received ranking number and the ranking number at the time of sending out of the data for a comparison is computed based on the element (data for a comparison) sent out in the sending-out processing by PMM of the first half, and the ranking number sent out in processing by PMM of the second half (step 3132-2). This difference "M" shows the number of the elements (that is, the ranking number smaller than the element concerned was attached) located ahead of the element used as the object for a comparison in the PMM group of the second half. Therefore, the same element as the element which serves as the object for a comparison concerned among the elements with which self holds each PMM which constitutes the PMM group of the second half is specified (step 3132-3), and in existing, it adds "M" to the ranking number of these elements, respectively (step 3134). After step 3134, PMM moves [number / of the elements concerned] a receiving pointer (step 3135). This processing is the same as that of step 3104 and abbreviation.
[0083] Next, explanation is added per [of calculation of the number of duplications

shown in drawing 24 , and the sorting application (a "sort body" is called by the case.) shown in drawing 31 R> 1] parallelism. As shown in drawing 23 , in the gestalt of this operation, buses 2301 and 2302 and 2303 grades are used for the communication link between PMM(s) which use buses 2304 and 2305 and start activation of a sort body in the communication link between PMM(s) concerning the count of the number of duplications. Then, if parallel processing is possible in PMM, the count and sort body of the number of duplications can be arranged in parallel and performed. In this case, it is if calculation of the number of duplications about a certain element is completed in the PMM group of the first half (for example, as shown in drawing 28 (a)). An element "****" and its number of duplications "2" are sent out to the 2nd bus, and if received by PMM (PMM1-PMM4) which constitutes the PMM group of the first half, processing as shown in drawing 3131 can perform about the element with which the number of duplications was computed. That is, calculation of the number of duplications of a certain element can be answered, and processing of step 3102 about the element concerned - step 3104, processing of step 3112 - step 3122, and processing of step 3132 - step 3135 can be performed. Moreover, although the element about a certain element, its number of duplications, etc. are related with the element concerned among the processings shown in above-mentioned drawing 31 , they can be deleted with termination. Therefore, in each of PMM which constitutes the PMM group of the first half, it is not necessary to hold all (for the number of different elements to follow the amount on increasing, and it to become large) of an element and the data about the number of duplications.
[0084] Thus, in the gestalt of the 2nd operation, in the PMM group of the first half, the number of duplications was computed and an element and its number of duplications are sent out to the PMM group of the second half. It becomes unnecessary therefore, for the PMM group of the first half to overlap and send the same element to the PMM group of the second half. When many same elements overlap especially (for example, that an element indicates man and woman's classification to be, the thing which shows age), it becomes possible to decrease the count of processing of a sort body, and sorting application can be realized more at a high speed.
[0085] Explanation is added per gestalt of [compile processing (gestalt of the 3rd operation)], next implementation of the 3rd of this invention. With the gestalt of the 3rd operation, the pointer array for specifying a record to a record, the value list which has arranged each element without duplication, and a value list based on the array which consists of an element arranged in each PMM is created. This processing is called compile in this description. For example, what is necessary is just to connect

PMM, as shown in drawing 32 when a certain array element is alloted by four PMM(s) (PMM1-PMM4). drawing 32 -- being shown -- as -- PMM -- one - PMM -- four -- an input/output terminal (I/O) -- the -- one -- a bus (sign 3201 reference) -- connecting -- having -- the -- on the other hand -- PM -- one - PMM -- four -- an output terminal -- (-- O --) -- and -- others -- PMM"k -- " -- an input terminal -- (-- I --) -- the -- two -- a bus (sign 3202 reference) -- connecting -- having -- **** .

[0086] The 1st bus is used for information interchange of the PMM group which consists of PMM1-PMM4, and the 2nd bus is used in order to give an element and its number of duplications to other PMM"k." In the gestalt of this operation, a value list, the number array of existence, etc. are formed in other PMM"k" based on the above-mentioned element and its number of duplications. In addition, although this PMM"k" may be PMM(s) other than PMM1 - PMM4, of course, it may be in any of PMM1-PMM4. Drawing 33 is a flow chart which shows the compile processing concerning the gestalt of this operation. In addition, in order to give explanation easy, as shown in drawing 34 (a), the element is alloted by PMM1-PMM4, and I think to them that processing which attaches a ranking number among these has already been performed. First, the ranking number counter which shows the ranking number of each value (element) applied to processing in PMM, The value number counter which shows the ranking number of the value concerned after processing (element), the same value number counter with which the element concerned shows whether which overlaps and it exists, And a value-preserving register is prepared last time holding the value (element) which became a processing object in the last processing, and initial value "0" is given to each counter (refer to step 3301 and drawing 34 (a)). In addition, a value is not held last time in first stage at a value-preserving register.

[0087] Processing of step 3302 of the following and drawing 33 - step 3306 is the same as that of steps 24021-2406 of drawing 24 , and abbreviation. That is, it judges whether each element to which PMM specified the ranking number with reference to the ranking number counter so that it might become a processing object, and the ranking number concerned was given is what self holds (step 3303). In the state of drawing 34 , since the counter value of a ranking number counter is "0", PMM3 creates the number of the existence in self-PMM (in this case, "1") which shows how many PMM3 concerned holds the element "****" to which the ranking number "0" was given to the 2nd bus (step 3306 reaching (referring to drawing 34 (b)).). Subsequently, when the element memorized by the value-preserving register last time is compared with the element emitted to the 1st bus and these are different, PMM3

assigns the counter value of a value number counter to the ranking number, as sent out to the 1st bus (step 3307). In addition, in the state of drawing 3434 , since the counter value of a value number counter is initial value "0", the ranking number concerning an element "****" does not change (refer to drawing 34 (b)).
[0088] Subsequently, each data given through the 1st bus in PMM is received (step 3308). Processing of steps 3308-3311 is the same as that of processing of steps 2408-2401 in drawing 24 , and abbreviation. namely, every -- PMM applies the number of the existence in PMM of the data given to the counter value of a ranking number counter, and further, when an element is not new among the given data (step 3310 no (No)), it applies the number of the existence in PMM to the counter value of the same value number counter (refer to step 3311 and drawing 34 (b)). As shown in drawing 34 , after the processing about the element "****" to which the ranking number "0" was given is completed, processing about the element to which the ranking number "1" was given is performed similarly (refer to drawing 35 (a)).
[0089] Furthermore, processing about the element to which the ranking number "2" was given is performed. Here, PMM1 compares the element "****" memorized by the value-preserving register last time with the element "******" to which the ranking number "2" was given. Here, since these are different (it is yes (Yes) at step 3304), PMM1 sends out the element memorized by the value-preserving register last time and the counter value of the same value number counter to the 2nd bus (step 305). Subsequently, PMM1 sends out the number of the existence in self-PMM "1" which is the number of the element "******" of a processing object, and the elements "******" which PMM1 holds to the 1st bus (step 3306). After that, PMM1 compares the element memorized by the value-preserving register last time with the element emitted to the 1st bus. Since these are different in emitting an element "******", the value (0+1=1) which added "1" at the counter value of a value number counter is assigned to the ranking number of an element "******." The number of the existence in self-PMM in the data which received each data with which PMM was given through the 1st bus (step 3308), and received it in the counter value of a ranking number counter is added (refer to step (2+1=3) 3309 and drawing 35 (b)). since the element "****" of a value-preserving register differs from the given element "******" last time when an element "******" is given (it is yes (Yes) at step 3310) -- every -- PMM performs new price registration processing (step 3312). In this processing, the increment (0+1=1) of the counter value of a value number counter is carried out, and the counter value of the same value number counter is changed into the number of the existence in self-PMM in the received data "1", and the content of the

value-preserving register is rewritten by the element "******" last time (refer to drawing 35 (b)).
[0090] Same processing is performed also about the element "******" of a ranking number "3." For example, PMM3 sends out an element "******" and the number of the existence in self-PMM "1" to the 1st bus (step 3306 reference), and assigns the counter value "1" of a value number counter to the ranking number of the element "******" concerned (refer to step 3307 and drawing 36 (a)). moreover, every -- the number of the existence in self-PMM "1" which received PMM to the counter value of a ranking number counter -- adding (step 3309 reference) -- the number of the existence in self-PMM "1" is applied to the counter value of the same value number counter (refer to step 3311 and drawing 36 (a)). furthermore, the element of a ranking number "4" -- "-- passing -- a basis -- ", as it is also related and is shown in drawing 3636 (b) PMM1 sends out the counter value "2" of an element "******" and the same value number counter to the 2nd bus (step 3305). the 1st bus -- an element -- "-- passing -- a basis -- " -- and the number of the existence in self-PMM "1" -- sending out (step 3306) -- and an element -- "-- passing -- a basis -- " -- the value (1+1=2) which added "1" at the value number counter is assigned to a ranking number. Also in one of these, and each PMM, count-up of a ranking number counter and new price registration processing are performed (refer to steps 3309 and 3312 and drawing 36 (b)). Same processing is performed also with the element of other ranking numbers. The processing about each element is shown in drawing 37 (a), (b), and drawing 38 . in addition, the drawing 3838 -- being related -- PMM1 -- the last element -- "-- passing -- a basis -- " -- and the data in which the number of existence of the element is sent out to the 1st bus, and termination is shown are outputted to the 2nd bus (step 3315 reference).
[0091] The input of PMM"k" is connected to the 2nd bus as mentioned above. Therefore, an element without duplication and the counter value of the value number counter about this are given to the 2nd bus. Therefore, PMM"k" receives these, and it carries out sequential arrangement of the counted value of the received value number counter at the number array of existence while it carries out sequential arrangement of the received element at a value list. Drawing 39 (a) is drawing showing the value list and the number array of existence which were created in PMM "k." These are sent out at step 3305 or step 3314 (refer to drawing 35 (b), drawing 36 R> 6 (b), and drawing 38), and are transmitted to PMM"k." As shown in drawing 39 (a), the number of existence (namely, the number of duplications) which shows how many it is arranged without duplication of an element at a value list, and each element exists is

arranged at the number array of existence.
[0092] Furthermore, in PMM1-PMM4, the pointer array to the value list to which a record and the ranking number attached so that there might be no duplication in each element are made to correspond can be created. That is, if the array to which the record and the ranking number given to the element corresponding to the record concerned were made to correspond is created, this can consider as the pointer array to a value list (refer to drawing 39 (b)). In drawing 39 (b), the ranking number "2" of a corresponding element serves as pointer value in the pointer array to a value list about a record "0." This shows what a value which is a storing location number "2" should be directed for in the value list (refer to drawing 39 (a)). That is, the value list stored in PMM"k" can be directed with the pointer value of the pointer array to a value list, and this becomes possible to specify an element from a record.
[0093] Thus, according to the gestalt of this operation, it is carrying out swing direct [of the ranking] so that the array element alloted by PMM is sorted, a ranking number may be attached and the same ranking may be given to the same element. An element is matched with ranking without the newly obtained duplication, and is stored in a value list. the element under array in which the ranking concerned was alloted as a pointer array to a value list -- a response -- the price -- ******** . Therefore, based on a record, it becomes possible through the pointer value of a pointer array to specify the element under value list.
[0094] Explanation is added per gestalt of [share-izing (gestalt of the 4th operation) of a value list], next implementation of the 4th of this invention. Two arrays are share-ized in the gestalt of the 4th operation (join). As this premise, the pointer array to the value list by compile processing and a value list is created. Moreover, Space ID was given to the pointer array to a value list and a value list, and each PMM grasps in it the various information about the array which self allots by the space ID concerned etc.
[0095] Drawing 40 is a flow chart which shows the share-ized processing concerning the gestalt of the 4th operation. In order to give explanation easy, as shown in drawing 41 (a), the array (sign 4100 reference) which consists of an element corresponding to a record as original data thinks that it is alloted by a certain PMM group. The block (a "information block" is called hereafter.) which becomes the PMM group which consists of PMM1 and PMM2 from a pointer array (sign 4101 reference) and a value list (sign 4102 reference) by compile processing about this record group is formed. I think that the information block to which other arrays (sign 4110 reference) which consist of the element corresponding to a record as original data are alloted by other

PMM groups, and become the PMM group which consists of a PMM group 3 and a PMM group 4 from a pointer array (sign 4111 reference) and a value list (sign 4112 reference) by compile processing on the other hand is formed.
[0096] The instruction which directs the join of a value list is transmitted to each PMM from CPU12 with the space ID of the array which shows two value lists. every -- a thing [as / whose array which should be carried out a join among PMM(s) is the value list which self holds, or its part] (that is, the above-mentioned example PMM1-PMM4) specifies the value list set as the object of a join based on Space ID (refer to step 4001 and drawing 42 (a)). Subsequently, PMM1-PMM4 are in the condition which joined two value lists, sort these and give a ranking number to each element (step 4002). The sorting application which starts the gestalt of the 1st operation for this sorting application can be used. In the above-mentioned example, the 1st PMM group which consists of PMM1 and PMM2, and the 2nd PMM group which consists of PMM3 and PMM4 are alike first, respectively, it sets, and processing of ranking numbering of an element is performed. subsequently A ranking number is given to the element in two PMM groups by making the 1st PMM group into the PMM group of the first half, and making the 2nd PMM group into the PMM group of the second half. Drawing 42 (b) is drawing showing the condition that did in this way and the ranking number was given to the element.
[0097] Compile processing is performed between PMM(s) which allot the value list used as a processing object after that, and the number array of existence which was communalized by any [other PMM(s) or] of PMM1-PMM4 they are by this and which was communalized [which were communalized and was value-listed] is generated (step 4003). That is, a new value list which the element of the merged value list does not overlap by compile processing, and the number array of existence in which each element stored the number of existence which shows which overlaps and exists are acquired (refer to drawing 42 (c)). The new pointer array for directing the new value list (namely, value list obtained by compile processing) by which the join was carried out after such processing is searched for. This is realized by creating a new pointer array which stores in the location of the pointer value under pointer array concerned, and a corresponding location the ranking number which the pointer value under pointer array in the information block before share-izing shows, and to which the ranking number array acquired by compile processing corresponds. Dealing with the new ranking number (referring to drawing 42 (c)) given to each element could understand the value under above-mentioned ranking number array.
[0098] As shown in drawing 43 (a), for example, since the value (ranking number)

under ranking number array of the location which the 1st pointer value under pointer array "1" shows is "2", the pointer value of the location where it corresponds under pointer array after being share-ized is set to "2." Moreover, since the value (ranking number) under ranking number array of the location which the 2nd pointer value "2" shows is "3", the pointer value of the location where it corresponds under share-ized pointer array is set to "3." Thus, it becomes possible to acquire the pointer array about the value list by which the join was carried out (drawing 43 (a) and drawing 43 (b)).
[0099] Probably, it will be clear that a value (element) can be specified from a record with such a new pointer array and the value list by which the join was carried out. As shown in drawing 44 , the pointer value of the location where a record corresponds during the pointer array to the newly obtained value list is specified, and the element under value list in the location which the value shows [the pointer value concerned] is specified. Here, that the same element as the element of the original data is specified could understand, in spite of carrying out the join of the two value lists.
[0100] Thus, according to the gestalt of the 4th operation, the ranking number array of the value list by which the join was carried out, and each value list is acquired by merging two or more value lists and combining sorting application and compile processing about the element of the merged value list. The value (ranking number) of a ranking number array is specified according to the pointer array for specifying a value list from a record, and the new pointer array for specifying the value list by which the join was carried out based on the record can be acquired by storing the ranking number concerned in the location corresponding to a record. Therefore, it becomes possible to carry out the join of two or more value lists with the time amount of the sorting application mentioned above, and time amount extent of compile processing, and it becomes possible to raise processing speed remarkably.
[0101] Various modification is possible for this invention within the limits of invention indicated by the claim, without being limited to the gestalt of the above operation, and it cannot be overemphasized that it is that by which they are also included within the limits of this invention. For example, in the gestalt of said operation, although it has applied to the computer system, it is not limited to this, and this invention can also be applied to a computer board connectable with a personal computer etc. In this case, in drawing 1 , CPU12, the memory unit 14, and bus 24 grade are carried on a board, and this constitutes the information processing unit in this invention.
[0102] Moreover, the number of the groups of the bus which connects between CPU12 and memory modules 14 and/or between memory modules 14 is not limited to

the gestalt of said operation, and can be suitably determined in consideration of the magnitude of the circuit board in which a computer system is carried, the number of bits of each bus, etc. Moreover, in the gestalt of said operation, it is between the switch 28 for specifying connection with I/O and bus of a memory module, and CPU and a memory module, between memory modules, or between I/O of a memory module, and the switch 30 from which a bus can be cut is formed. forming switches 29 and 30 -- for example, while using a certain bus (24 to bus 4 reference of drawing 1) for data transfer with the CPU module 12 and a memory module 14-1, it can be simultaneously used for the data transfer between a memory module 14-2 and a memory module 14-3 (in this case, what is necessary is just to turn OFF a switch 30-5). Therefore, it is possible to use a bus for validity more. However, when the group of a bus can enlarge a number enough, or when there are comparatively few memory modules, it is not necessary to necessarily prepare a switch 29 or 30.
[0103] Moreover, in this description, although it indicated that the instruction from CPU12 was given through the control signal line 25, it cannot be overemphasized that various control signals, such as a clock besides an instruction, for each memory module to synchronize and operate are given through the control signal line 25, and the predetermined signal (for example, an error signal and the signal which shows data acceptance) from each memory module to CPU12 is given.
[0104] Furthermore, in the gestalt of said operation, although the various connection between PMM(s) was illustrated, selection of the bus used for the connection and transmission and reception between PMM(s) is not limited to what is shown in the gestalt of the above-mentioned implementation.
[0105] Moreover, in the gestalt of said 3rd operation, as shown in drawing 32 , the 1st bus (sign 3201) is used. It is not what is limited to this although the number of existence of an element and the element concerned (the number of duplications) is communicating the communication link between each PMM using nothing and 2nd refer to the bus (sign 3202). For example, as shown in drawing 45 , PMM"k" which generates the value list which is the array of an element without duplication, and its number array of existence acts as the monitor of the 1st bus 4501. Based on the element and the number array of existence which appear on the 1st bus 4501, predetermined processing (for example, maintenance/renewal of count-up of a counter or the content of the register performed in PMM1-PMM4) may be performed, and a value list and the number array of existence may be created.
[0106] Furthermore, in this description, even if the function of one means is realized by two or more physical means, the function of two or more means may be realized by

one physical means.
[0107]
[Effect of the Invention] According to this invention, it becomes possible remarkably by the stable processing time to offer the information processor in which the sort of an array, compile, and a join are possible at high speed.

[Translation done.]

TECHNICAL FIELD

[Industrial Application] This invention relates to a detail extremely more about the information processor of a distributed memory type at the information processor which can realize sort, compile, and processing of a join at a high speed.

[Translation done.]

PRIOR ART

[Description of the Prior Art] A computer is introduced into various locations of the whole society, and large-scale data came to be stored here [there] by the end of today when networks including the Internet permeated. In order to process such large-scale data, huge count is required, therefore trying to introduce parallel processing is natural.
[0003] Now, parallel processing architecture is divided roughly into a "shared memory mold" and a "distributed memory type." The former ("shared memory mold") is a method with which two or more processors share one huge room. Since the traffic between a processor group and a shared memory serves as a bottleneck by this method, it is not easy to build a realistic system using the processor exceeding 100. In case the square root of 1 billion floating point variables is calculated by following, the acceleration ratio to a single CPU will call it at most 100 times. Experientially, about 30 times is an upper limit.

[0004] Each processor has respectively local memory, and the latter ("distributed memory type") combines these, and builds a system. The design of the hardware system which also incorporated hundreds - tens of thousands of processors by this method is possible. Therefore, it is possible to make the acceleration ratio to the single CPU at the time of calculating the square root of the 1 billion above-mentioned floating point variables one 10,000 times the number [hundreds -] of this.

[Translation done.]

EFFECT OF THE INVENTION

[Effect of the Invention] According to this invention, it becomes possible remarkably by the stable processing time to offer the information processor in which the sort of an array, compile, and a join are possible at high speed.

[Translation done.]

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] Although it is said that the potential need of the parallel processing by the processor of a large number exceeding hundreds is large, if a current realistic hardware technique tends to realize this as mentioned above, the design by technique other than a distributed memory type is difficult. In a distributed memory type, since the capacity of the memory attached to each processor is small, in the maintenance and processing of large-scale data (usually array) which are one of the main objects of parallel processing, the memory attached to two or more processors and each needs to allot this.
[0006] However, when the memory attached to two or more processors and each allots an array, the bus mediation for preventing the collision of the data on a bus is difficult, if each processor cannot operate in juxtaposition, utilization effectiveness of

a processor cannot be improved, consequently there is a trouble of being unable to attain improvement in the speed of processing. Then, this invention attains the various objects as follows.

[0007] (1) The collision of the data on a bus does not occur in algorithm, but bus mediation is unnecessary and, thereby, raise processing speed taking advantage of the bandwidth of a bus fully.

(2) Enable it to assign the processing which made parallel processing by these possible, and used each memory module effectively combining many memory modules equipped with the processor (two or more desirable processors) and the graduation, and became independent to the processor in each memory module, and, thereby, raise processing speed further by deployment of a memory module.

(3) When magnitude of the data for a sort is set to "N", need only the magnitude of the data of O (N). (In the conventional sorting application, when the worst, the amount of data of O (N*N) or O (N*Log (N)) may be needed.)

(4) The processing time is stable, and even when the worst, expectable processing speed is guaranteed. That is, remarkably, this invention is the stable processing time and aims high-speed at offering the information processor which can sort an array.

[Translation done.]

MEANS

[Means for Solving the Problem] The connection between two or more memory modules with which, as for the object of this invention, a CPU module and each have MPU and a RAM core, and said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU Are the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module, and the sort of the element with which MPU of said memory module constitutes the part of the array which self grasps is performed. The location which said part which said self grasps as the sort means which rearranges said element according to specific sequence occupies during an array is followed. Said sorted element with the ranking number It transmits to other

memory modules through a predetermined bus. By the comparison with I/O which receives other said elements and ranking numbers from a memory module through a predetermined bus, and the element which self grasps when said element and a ranking number are received A ranking number calculation means to compute the virtual ranking number which is the candidate of the ranking number of the received element, and to return a memory module besides the above, The presentation memory module of the side which is equipped with a ranking decision means to decide the ranking of an element, according to the virtual ranking number concerned, and sends out said element and a ranking number when said virtual ranking number is received, It is attained by the information processing system characterized by deciding the ranking number of said array element by the communication link with said element and the near judgment memory module which computes a virtual ranking number by carrying out ranking number acceptance.
[0009] According to this invention, it performs through a bus with presentation of the element by the presentation memory module, and a ranking number, a virtual ranking number is computed with a judgment memory module, and the virtual ranking number concerned is given to a presentation memory module through other buses. Therefore, in a presentation memory module and a judgment memory module, it becomes possible to be able to advance sorting application in juxtaposition and to also avoid the collision of a bus.
[0010] An element specification / sending-out means to specify the element with which said memory module serves as a processing object in the desirable embodiment of this invention according to the settled ranking number, and to send out to which bus, When the same element as an element comparison means to compare the element used as the last processing object with the sent-out element is sent out It has the same value number counter which counts up the value and which shows the number of existence of the same element. When it is judged that the element with which said element comparison means became the last processing object differs from the sent-out element It is constituted so that the value of the same value number counter about the element used as the last processing object and the element concerned may be associated and any may send out. Furthermore, it has the array arranged in the sequence which received the value of the element with which which memory module became the sent-out last processing object, and a related counter, and associated and received these. According to this embodiment, in which memory module, an element and its number of duplications are received in predetermined sequence, and become possible [that this creates the array of an element without

duplication, and the array which shows the number of existence of each element]. That is, the number with which each element exists in the list of elements without duplication and the array of a basis by this can be grasped easily.
[0011] According to another embodiment of this invention, said memory module said element comparison means The value number counter which counts up the value when it is judged that the element used as the last processing object differs from the sent-out element and which shows a ranking number without duplication, When the element used as the last processing object and the sent-out element are the same, about the sent-out element the value of a value number counter It was decided that it would be the ranking number of the element concerned without duplication, on the other hand, the value of the value number counter counted up when [that] these differed was determined as the ranking number of the element concerned without duplication, and it has a renewal means of a ranking number to update the ranking number concerned. According to this embodiment, it becomes possible to change the ranking number given to the array element into the thing in the condition of having eliminated duplication of an element.
[0012] Moreover, two or more memory modules with which, as for the object of this invention, a CPU module and each have MPU and a RAM core, The connection with said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU Are the sort approach of an array of having used the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module, and it sets to the (a) memory module. The step which sorts the element which constitutes the part of the array which self grasps, (b) The location which said part which said self grasps occupies during an array is followed. The step which determines the judgment memory module of the side which receives the presentation memory module, element, and ranking number of the side which sends out an element and a ranking number among the memory modules which grasp the part of said array, (c) In a presentation memory module, the sorted element with the ranking number In the step transmitted to other memory modules through a predetermined bus, and (d) judging memory module In the step which receives other said elements and ranking numbers from a memory module through a predetermined bus, and the (e) aforementioned judging memory module Based on the ranking number of the element which the judgment memory module concerned grasps, the virtual ranking number which shows the candidate of the

ranking number of the received element is computed. In the step which returns the virtual ranking number concerned to said presentation memory module, and the (f) aforementioned presentation memory module The step which updates the ranking number of an element according to the virtual ranking number concerned when said virtual ranking number is received, (g) whenever said step (d) - (f) is completed, the memory module group which consists of the presentation memory module and judgment memory module about the element to which the predetermined ranking number was given by concerned step (d) - (f) By repeating step (d) - (f) as either a presentation memory module group or a judgment memory module group, respectively By updating the ranking number of the element in each memory module group, it is attained also by the sort approach characterized by deciding the ranking number of each element of an array.
[0013] According to the above-mentioned invention, sending out of sending out of the operation in a presentation memory module, the element in a presentation memory module, and a ranking number, the operation in a judgment memory module group, and the virtual ranking number in a presentation memory module can perform in juxtaposition, and can also avoid the collision of a bus. That is, this becomes possible remarkably to realize sorting application (ranking numbering to an array element) at a high speed. Moreover, the amount of memory to be used also becomes possible [controlling to O (N)]. In the desirable embodiment of the above-mentioned invention, a step (e) contains the step which computes a virtual ranking number based on the number of front insertion which shows the number of the elements which should be located ahead, the ranking number about the element which should be located ahead, and the received ranking number from the element received (e1). Furthermore, in a desirable embodiment, a step (f) contains the step which substitutes the virtual ranking number received (f1) for the ranking number of the element sent out at the step (c).
[0014] In the desirable embodiment of this invention, it sets in (h) presentation memory module group further. The element grasped with the memory module which constitutes the presentation memory module group concerned So that it may have the step which computes the number of duplications which shows whether it has set in the memory module group concerned, and shoes existence is recognized and it may not transmit [said step (c) overlaps the same (c1) element and] it The number of front insertion which shows the number of the elements which should be ahead located from the element with which said step (e) received the sorted element with the ranking number and the number of duplications including the step transmitted to

other memory modules (e2), The step which computes a virtual ranking number is included based on the ranking number about the element which should be located ahead, and the ranking number and the number of duplications which were received. Said step (f) A virtual (f2) ranking number, Based on a difference with the ranking number at the time of sending out of the element in a step (c), the step which determines the ranking number of the same element as the element concerned is included.
[0015] According to this embodiment, a presentation memory module does not need to send out the same element repeatedly. Moreover, if the number of duplications of a certain element is computed, the ranking number and the number of duplications of the element concerned can be transmitted to a judgment memory module, and calculation of the virtual ranking number concerning the element concerned can be performed in a judgment memory module. That is, thereby, it can prevent lowering the utilization effectiveness of a memory module.
[0016] In a still more desirable embodiment, the presentation memory module group which consists of a memory module which is 2n to which the increment of the n (n:1 or more integers) is carried out whenever a presentation memory module is an independent memory module, and a receiving module is also an independent memory module and the step of (d) - (f) ends it, and the judgment memory module group which consists of a 2n memory module are formed in first stage. If a 2n memory module is used as mentioned above, it will become suitably possible to realize sorting application.
[0017] Moreover, in another embodiment of this invention, an array is sorted by the above-mentioned sort approach. Based on the sorted array concerned, and the compile approach that the element under said array generates the new array arranged in predetermined sequence that there is no duplication (i) The step which sends out the element which serves as a processing object according to a ranking number in a predetermined memory module, (j) when the same element as the element used as the last processing object is sent out Count up the same value number counter which shows the number of existence of the same element, and on the other hand, when the element used as the last processing object and a different element are sent out The step which associates the value of the same value number counter about the element used as the last processing object, and the element concerned, and sends these out, (k) by receiving the value of the element used as the last processing object, and the same related value counter, having the step which associates these and is arranged during a new array, and repeating (l) step (i) - (j) It is characterized by associating an element and its number of existence, and being arranged during said new array.

[0018] moreover, the above-mentioned compile approach -- further -- (m) -- in which module, it has the step which acts as the monitor of the value of the element sent out at a step (j), and the same related value number counter, and a step (k) may be performed with which the module concerned.
[0019] In the memory module which grasps the array element concerned moreover, the above-mentioned compile approach -- (n) -- While forming the ranking number counter and the same value number counter which store the ranking number of the element used as a processing object, and the number of existence of the element concerned, respectively And it sets to the memory module which grasps the step which prepares the register which stores the element used as the last processing object temporarily, and the element to which the ranking number concerned was given according to (o) ranking number. When the step which sends out the element concerned to the 1st bus, and the element received in the memory module which grasps the (p) array element are compared with the content of the register and these are in agreement While counting up the number of existence, when these are not in agreement In which memory module the step which updates the content of the register, and the value of the number counter of existence after sending out the content of the register, and the value of the number counter of existence to the 2nd bus, and (q) -- You may have the step which arranges the content of said register, and the value of the number counter of existence during an array as the number of existence of an element and the element concerned, respectively.
[0020] A step (n) contains the step which forms the value counter which stores the ranking number which does not have duplication about the element used as a processing object (n1) further. Said step (p) (p1) When the received element is compared with the content of the register and these are in agreement While giving the value of a value number counter to the ranking number of the element used as the processing object concerned When these are not in agreement, it is still more desirable to include the step which gives the value of the value number counter which counted up the value number counter and was counted up for the ranking number of the element used as a processing object.
[0021] In another embodiment of this invention, moreover, the join approach of an array of realizing share-ization of two or more arrays, using the above-mentioned sort approach and the above-mentioned compile approach (r) The step which performs processing concerning said sort approach which joins two or more arrays and gives a ranking number to each of these array elements, (s) According to the element and its ranking number under said array which joined, it has the step which generates the new

array in which processing concerning said compile approach is performed and the duplicate element does not exist. That is, it becomes possible by giving the sort approach and the compile approach concerning this invention, where a desired array is merged to acquire the array which eliminated duplication of an element and by which the join was carried out.
[0022] Two or more memory modules with which a CPU module and each have MPU and a RAM core in still more nearly another embodiment, The connection with said CPU and memory module, and/ Or it has two or more sets of buses which connect between the memory modules of arbitration. The processing about the array grasped with said one or more memory modules by the instruction given to MPU of each memory module from CPU The join approach of two or more arrays of having used the distributed memory-type information processing system constituted so that it might perform by actuation of MPU of each memory module is [0023]. In order to specify the predetermined element in the value list which is the array in which the above-mentioned sort approach and the above-mentioned compile approach are used for, and the memory module stored the element based on the record number, respectively Equip the location corresponding to a record number with the pointer array which has arranged the pointer value which shows a value list, and two or more (r1) value lists are joined. The step which performs processing concerning said sort approach which gives a ranking number to each of these array elements, (t) While generating the new value list with which processing concerning said compile approach is performed, and the duplicate element does not exist according to the element and its ranking number under said value list which joined It has the step which considers the array which consists of a ranking number of the step which updates the ranking number of said element for the ranking number of the element concerned in case the duplicate element does not exist, and an element in case the element which carried out the (u) aforementioned duplication does not exist as a new pointer array to show a new value list.
[0024]
[Embodiment of the Invention] With reference to an accompanying drawing, explanation is added per gestalt of operation of this invention below a [hardware configuration]. Drawing 1 is a block diagram which shows the configuration of the computer system concerning the gestalt of operation of this invention. As shown in drawing 1 R> 1, a computer system 10 The CPU module 12 which realizes parallel operation by single instruction, the memory module 14-1 which memorizes various data required for parallel operation, 14-2, 14-3, and --, It has the fixed memory 16

which memorizes a required program and data, the input devices 18, such as a keyboard and a mouse, the indicating equipment 20 which consists of CRT etc., and the legacy memory 22 the data of various formats etc. are remembered to be. Moreover, in a bus 24-1, 24-2, and --, transfer of the information between the circuit elements which a switch 28-1, 28-2, 28-3, --, etc. were arranged by the contact with the CPU module 12 and each memory module 14, and were chosen as it is possible. Moreover, the switch 30-1 for making connection and connection of a bus, 30-2, and -- are prepared between adjoining memory modules between the CPU module 12 and the memory module 14-1. Moreover, the switch (sign 29 reference) may be formed between the contact of the input terminal of a memory module, and a bus, and the contact of the output terminal of the memory module concerned, and a bus. The above-mentioned switch is shown by the round mark of a broken line in drawing 1 .
[0025] Furthermore, it is desirable to prepare a single input terminal and not only a single output terminal but other one or more terminals (input/output terminal etc.) in the memory module 14. For example, in the gestalt of the 2nd operation and the gestalt of the 3rd operation which are mentioned later, processing is realized using the I/O from three or more terminals.
[0026] Between the CPU module 12 and the memory module 14, two or more buses 24-1, 24-2, 24-3, 24-4, and -- are prepared. Therefore, transfer of data etc. is possible for between the CPU module 12 and a memory module 14 and between memory modules by the above-mentioned bus. Moreover, the control signal line 25 is formed between CPU12 and a memory module 14, and the instruction emitted from CPU12 is transmitted to all the memory modules 14.
[0027] Furthermore, the local bus 26 is arranged between CPU12 and other components, and transfer of data etc. is possible for for example, fixed memory 16, an input device 18, etc. also among these. CPU12 reads the program memorized by other storage (not shown) like RAM which was memorized by fixed memory 16 or was connected on the bus 26, and performs control of the switches 28-30 besides transfer of data including sending out of the instruction to the memory module 14 shown below etc. according to this program. Moreover, CPU12 can receive the data of the various formats memorized by the legacy memory 22 according to a program, can change them into a series of data (array) which can process the data of this format by the system which consists of CPU12, a memory module 14, and a bus 24, and can also store these in each memory module 14.
[0028] Drawing 2 is a block diagram which shows the outline of each memory module 14. As shown in drawing 2 , a memory module 14 The clock buffer 32 which accepts

synchronizing signals, such as a clock given from the CPU module 12, The space ID mentioned later, the element number of data, etc. are grasped as the RAM core 34 which memorizes data. MPU36 which controls data read-out from the data writing and RAM core to the RAM core 34 based on Space ID and an element number when the instruction from CPU12 etc. is received, It has I/O38 which receives the data from either of the buses, and supplies the RAM core 34, and/or sends out the data from the RAM core 34 to which bus. In the gestalt of this operation, through the control signal line 25, a memory module 14 accepts the instruction from CPU, MPU36 can answer this instruction, and can read the data of the RAM core 34, and it can write data in the RAM core 34, or can perform predetermined processing now to data. Moreover, a data input and data output are performed based on synchronizing signals, such as a clock given to the clock buffer 32, through the data access to the RAM core 34, and I/O. As for MPU36 of the above-mentioned memory module 14, it is desirable that it consists of two or more processing units, and two or more processings can be performed in juxtaposition.
[0029] In this invention, it is possible that a computer system 10 is a system of a memory share mold so that clearly from drawing 1 and drawing 2 . Moreover, each memory module 14 performs processing in juxtaposition by giving an instruction to each memory module 14 through the control signal line 25 so that it may mention later. Moreover, the data output to a bus, the data input from a bus, etc. are performed based on a predetermined synchronizing signal. Therefore, it is possible that this computer system 10 is making the gestalt of SIMD. Fundamentally, the computer system 10 equipped with such a configuration is equipped with the multi-space memory concerning a design of this invention person indicated by Japanese Patent Application No. No. 263793 [11 to], the memory module, and the rearrangeable bus. Lessons is taken from these and explanation is simply added to below.
[0030] (1) In a multi-space memory book description, multi-space memory means the room assigned in order to access room based on Space ID and the address. Thereby, even if a series of data are alloted by many processors, each processor can separate and recognize this certainly.
[0031] In the conventional room, even if it might assign the field according to individual for every process, assigning room to every [a series of] variables (an array, structure, etc.) was not performed. Therefore, such conventional room is hereafter called "single room." In the system of single room, since data are accessed only using the address, a series of data which have relation were not able to be separated, and it has not recognized. For this reason, even if parallel processing was actually possible, that

propriety was not able to be judged in many cases. Moreover, the garbage collection needed to be performed in order to secure the hold location of a series of data concerned, when making a series of new data hold in a certain single room.
[0032] On the other hand, in this invention, Space ID was introduced into room and the same ID is given to it about a series of data. Moreover, in a memory module 14, the space ID about the data currently held at the own RAM core 34 can be grasped, and, thereby, each [memory module 14 / itself] can determine the right or wrong of self actuation by referring to the space ID of the data accessed now. Moreover, since each memory module relates with Space ID and all or some of a series of data can be held, a certain data of a series of can be divided and stored in two or more memory modules 14, and, thereby, a garbage collection can be made unnecessary.
[0033] (2) In a memory module and this invention, each memory module 14 had MPU36, and grasp each element number of a series of data which self besides the above-mentioned space ID holds. Therefore, after receiving the instruction from CPU12, the data which MPU36 should access according to an instruction can judge whether it is what is held in the RAM core 34 of self, and can determine the right or wrong of the need as access. Furthermore, each memory module 14 is able to determine the assignment range of the tacit processing in the instruction in SIMD from the range of the suffix of the array element stored in the RAM core 34 of self. The storage sequence of the element which should be processed is replaced according to the instruction from CPU12, and each memory module 14 can sort the element currently held in the RAM core 34 of self.
[0034] (3) Pipeline processing is realized by setting to rearrangeable bus this invention, and CPU's12 turning on / turning off selectively a switch 28-1, 28-2, -- and a switch 30-1, 30-2, and --, and specifying the memory module 14 which should deliver and receive data. As shown in drawing 3 , for example, the data outputted from certain memory module 14-i other memory module 14-j -- giving -- and -- being concerned -- others, when the data outputted from memory module 14-j should be transmitted to memory module 14-k of further others CPU12 sets up the condition of each switch so that bus 24-m may be assigned for memory module 14-i and 14-j and bus 24-n may be assigned for memory module 14-j and 14-k.
[0035] Furthermore, not only when connection between single memory modules realizes, but these pipeline processing can be realized by connection between two or more of a series of memory modules (memory module group). According to the processing which it is going to attain, between each memory module can be switched, and a communication link can be schedule-ized so that the capacity of a bus can be

used for an one direction about 100% by carrying out a continuation transfer in the sequence that the data of the defined class were able to be defined, for every connection path. Thereby, the lowness of the performance of interprocessor communication which was the biggest problem of the parallel processing system of a distributed memory type is cancelable.
[0036] [Multi-space memory] Explanation is again added more to a detail about the memory management of each memory module in the computer system concerning this invention using multi-space memory, and the memory access according to an instruction. Drawing 4 is drawing for explaining the structure of a memory module 14 under multi-space memory. As shown in drawing 4 (a), a space ID managed table is prepared in the RAM core 34 in a memory module 14. Thereby, MPU36 of a memory module 14 becomes possible [grasping required information, such as the space ID of the data which self holds,].
[0037] As shown in drawing 4 (b), the logic starting address of a data constellation under the management of Space ID and CPU for every data constellation which self holds, the size of the field where the data constellation was assigned, the physical starting address in the RAM core 34, the total size of a series of data which have the space ID concerned, and the access-restriction flag that shows access restriction are stored in the space ID managed table. in the gestalt of this operation, reading appearance of the access-restriction flag is carried out, and a chisel is possible for it -- only (R) and writing are possible -- (R) and R/W are possible -- three conditions of (RW) can be shown now. When the data constellation which has a certain space ID is given, MPU36 of a memory module 14 finds out one or more fields which should hold the data constellation concerned into the RAM core 34, divides a data constellation into remaining as it is or 2 or more, and holds it in the field concerned. In this case, the logic starting address and allotment area size in the RAM core which held data actually with the given space ID, a logic starting address, total size, and an access-restriction flag are also memorized by the space ID managed table. Drawing 4 (c) is drawing showing the data in the RAM core 36 according to the space ID managed table by drawing 4 (b).
[0038] Explanation is added to below per [to [the outline of memory access], thus the constituted memory module 14] access. As shown in drawing 5 , CPU12 transmits a required instruction (for example, writing and read-out of data) to all the memory modules 14 through the control signal line 25 first at Space ID and the logical address, and a list. In each memory module 14, this is answered, the space comparator 52 formed in MPU36 compares Space ID with the space ID currently held on the space ID

managed table of self, and it judges whether self holds the same thing, and a address comparator 54 makes the same judgment about the logical address. Subsequently, when it is judged that the data with which MPU36 of a memory module 14 serves as a processing object by the instruction at the RAM core 34 of self are held, with reference to a space ID managed table, address KARIKYURETA 56 computes the physical address in the RAM core 34, and specifies the data used as a processing object. Thus, after data are specified, MPU36 performs processing (for example, writing and read-out of data) according to the instruction given from CPU12, and when required, it transmits data to CPU12 (refer to drawing 5 (c)).
[0039] [Sorting application (gestalt of the 1st operation)] Explanation is added per [concerning the computer system 10 constituted in this way] sorting application. In addition, in the following explanation, since the memory module concerning this invention is a memory module equipped with MPU (processor), PMM (Processor Memory Module) is called.
[0040] In order to make an understanding easy, as shown in drawing 6 , four PMM(s) consider the case where two elements (family name) are held, respectively. As shown in drawing 6 (a), the family name "****" whose suffix (namely, record number) of an element is "0", and the family name "****" whose suffix is "1" are held at a certain PMM (1st PMM 14-1). the family name "******" whose suffix is "2" at 2nd PMM 14-2, and a suffix are "3" -- "-- passing -- a basis -- " -- ** -- the family name to say is held. Hereafter, the family name corresponding to a suffix as shown at drawing 6 (a) is held also at 3rd PMM 14-3 and 4th PMM 14-4, respectively. The same space ID was given to the array which consists of these elements, and each MPU36 of PMM has managed in it a suffix (record number), a physical address stored actually of the element which the RAM core 34 of self manages using the space ID managed table.
[0041] For example, the instruction which sorts the array which has this space ID through the control signal line 25 from CPU12 thinks that it was given to each PMM 14-1 to 14-4. Drawing 7 is a flow chart which shows the procedure of the sorting application concerning the gestalt of this operation. If an instruction (for example, instruction "sort the element under array which has a certain space ID") is published by CPU12 as shown in drawing 7 (step 700) This instruction is answered and it sets to each PMM. Each MPU36 of PMM The instruction given through the control signal line 25 is received. The content is interpreted (step 701), the "space ID" in an instruction is investigated (step 702), and it judges whether it relates to the space ID of the data which the RAM core 34 of self holds (step 703). When judged as a no (No) at step 703, processing is ended, and when [that] judged yes (Yes), on the other hand, whether

the writing of the data constellation about the space ID concerned of MPU36 being attained with reference to the space ID managed table and a required check are performed (step 704). When it is judged by the check that it is abnormal (it is yes (Yes) at step 705), MPU36 notifies to CPU12 that the error arose through the control signal line 25. On the other hand, in [that] being normal, MPU36 performs the sorting application body described below (707 or less step).
[0042] First, each of PMM 14-1 to 14-4 relevant to processing performs the sort of the element which self holds (step 707). This sort is actually accompanied by exchange of the element in each PMM14. More specifically, MPU36 sorts the element held in the RAM core 34 of self using the known sort technique, such as quick sort. Drawing 6 (b) is drawing showing the condition that the element under array in each PMM shown in drawing 6 (a) was sorted. In addition, as shown in drawing 6 (b), it should care about that arrangement of the suffix (record number) of each element is also changed with the sort of the above-mentioned element.
[0043] subsequently, every -- only the number of the elements under array which self has held / managed secures the field (ranking number field) for arranging a ranking number, and, as for MPU36 of PMM14, gives the initial value of each ranking number (step 708). Drawing 6 (c) is drawing showing each condition that the initial value of a ranking number was given about PMM. Thus, a ranking number is given in first stage within the element sorted within each module. Subsequently, merge between adjoining pairs and ranking numbering are performed (step 709). In step 709, first, CPU12 controls the switches 28 and 30 on a bus 24, and connects one output and the input of another side to one input of a predetermined pair, the output of another side, and a list among PMM(s) relevant to sorting application. Also when [adjoining / two] it does not PMM and adjoin, as for the above-mentioned pair, it is desirable to consist of two PMM(s) located in near. For example, in drawing 1 , when the thing relevant to sorting application is PMM 14-1 to 14-4, it is desirable to make PMM 14-3 and 14-4 into a pair by making PMM 14-1 and 14-2 into a pair. As shown in drawing 8 , CPU12 so that the output of PMM 14-1 and PMM 14-2 may be connected to a bus 24-1 And so that the input of PMM 14-1 and the output of PMM 14-2 may be connected to a bus 24-2 A switch 28 is controlled to connect the input of PMM 14-3, and the output of PMM 14-4 to a bus 24-2 to control a switch 28 and to connect the output of PMM 14-3, and PMM 14-4 to a bus 24-1. Furthermore, CPU12 turns OFF further the bus 24-1 arranged between PMM 14-2 and PMM 14-3, the switch 30-5 on 24-2, and 30-6. In drawing 8 , the condition that what is expressed with the black dot has flowed is shown, and the condition that what is expressed with a circle [white] is not

connected with a flow thru/or PMM is shown. Moreover, other things follow the condition of other PMM(s) (not shown). In addition, in the example of drawing 8 , having divided by turning a bus 24-1 and 24-2 a switch 30-5, and turning OFF 30-6, and using the bus for validity more could understand.
[0044] Thus, if connection between PMM(s) is prescribed by CPU12 as typically shown in drawing 9 , the processing body of ranking numbering between the pairs of PMM will be performed. Drawing 10 thru/or drawing 12 are drawings showing typically ranking numbering about the array shown by drawing 6 , in order to make an understanding easy, and it is a flow chart which shows the ranking number processing between the pairs of PMM with more common drawing 13 . In drawing 10 R> 0 thru/or drawing 12 , although only the processing process in PMM 14-1 and PMM 14-2 was shown, processing in PMM 14-3 and PMM 14-4 is also performed in juxtaposition. In addition, in processing, what gives data first to PMM of another side is called PMM of the first half, and what is received (PMM of another side) is called PMM of the second half here. Since PMM of the first half presents an element and a ranking number, it can be called presentation PMM, and since it judges the ranking number shown PMM of the second half on the other hand, it can be called judgment PMM. Which PMM may turn into PMM of the first half among pairs. In this example, for convenience, PMM 14-1 turns into PMM of the first half, and PMM 14-2 is PMM of the second half.
[0045] First, in PMM of the first half, the pointer (a "PUT pointer" is called hereafter) in which a processing location is shown is arranged to an initial position (it sets into the part of the sorted array and is a head, i.e., the location of the "0th" watch). On the other hand, the element received from PMM of the first half and the pointer (a "comparison pointer" is called hereafter) in which the location which should be compared first is shown are arranged to an initial position (it sets into the part of the sorted array and is a head, i.e., the location of the "0th" watch) so that it may explain below in PMM of the second half (drawing 10 (a) and step 1301 of drawing 13 , 1311 reference). In the gestalt of this operation, the comparison pointer used in PMM of the second half has taken the gestalt of the array of structures (X, Y, Z). X shows the head location (that is, an "unsettled location" is called the head location of a non-compared element, and henceforth) which should be compared here. Y The total of the element received from PMM of the first half is shown ("the number of front insertion" is hereafter called by the case.). Z is the proposal (a "virtual ranking number" is hereafter called by the case.) of the ranking number of the element given from PMM of the first half in the imagination array which merged PMM of the first half, and PMM of the second half, and was acquired. It is shown.

[0046] Subsequently, the first data transfer is performed by MPU of PMM of the first half. In this data transfer, the element of the location which a PUT pointer shows is transmitted to PMM of the second half through a bus (drawing 10 (b) and step 1303, 1312 reference). In addition, in branching of step 1302, although always judged yes (Yes) in processing between two PMM(s), about this, it mentions later. In the first data transfer, an element "****" is transmitted to PMM of the second half. In PMM of the second half, the location which should insert the transmitted element "****" is discovered in the part of the array stored in PMM of the second half (step 1313). Actually, this should just discover the location which should be inserted rather than inserts a value. In the gestalt of this operation, the element stored in the part of each array of PMM is arranged in the condition of having sorted actually. Therefore, retrieval of an insertion point is realizable using the high-speed search technique, such as the BAISE cushion method (split half method). By discovering an insertion point, it is the element which ranking has not decided and it becomes possible to pinpoint the range of the element ahead located from an insertion point (for "the range 1" to be called hereafter). In addition, in the gestalt of this operation, when there is the same element, the agreement that the ranking of PMM of the first half has priority is carried out. Therefore, when the element "****" transmitted from PMM of the first half exists also in PMM of the second half, the ranking of the direction stored in PMM in the first half is that priority is given (that is, smaller ranking number).
[0047] In this example, it turns out that the element "****" transmitted from PMM of the first half is located ahead of an element "****" among the part of the array which PMM of the second half grasps, and it turns out that the element belonging to the range "1" does not exist by this (it sets to step 1314 and is [refer to drawing 10 (c) and] yes (Yes)). Then, MPU of PMM of the second half returns "0 (namely, head)" to PMM of the first half through the bus of another side as a ranking number of the transmitted element "****" (step 1315). Subsequently, MPU of PMM of the second half increments a virtual ranking number, and sets it to "1" while it increments the number of front insertion and sets it to "1" (step 1316). It is because it is necessary to increment the number of front insertion and since one element transmitted from front PMM increased this, and the ranking number of the following element needs to increment what was given at least this time (in this case, "0"). If the ranking number (insertion point) of an element is given from PMM of the second half (step 1332), MPU of PMM of the first half will store the given ranking number as a ranking number of the corresponding element (step 1334), and, subsequently will increment a PUT pointer (drawing 11 (a) and step 1335 reference). Thus, the ranking of a certain element in

PMM of the first half is decided.
[0048] Next, MPU of PMM of the first half transmits the element "****" of the location which a PUT pointer shows to PMM of the second half through a bus (drawing 11 (b) and step 1303 reference). In PMM of the second half, the location which should insert the transmitted element "****" is previously discovered like the time of an element "****" being transmitted (step 1313). It turns out that an element "****" is located behind an element "** et al." among the part of the array which PMM of the second half grasps (it is yes (Yes) at drawing 11 (c) and step 1314). Thereby, in the part of the array which PMM of the second half grasps, the ranking of each element can be decided in the number of the elements located an element "** et al." and ahead [its], and a list. MPU of PMM of the second half makes a detail decide the ranking of the above-mentioned element in the following procedures more.
[0049] First, the number of front insertion "Y" is applied to the ranking number about the element contained in the range "1", respectively (step 1317). Thereby, the ranking of the element contained in the range "1" is decided. In the example mentioned above, "0+1=1" and the ranking number of an element "** et al." are set to "1+1=2" by the ranking number of an element "****." Subsequently, an unsettled location is changed into the location of the next element of the element at the tail end in the range 1 while the ranking number of the element at the tail end is substituted for a virtual number among the elements contained in the range 1 (step 1318) (step 1319). In the above-mentioned example, the ranking number "2" of an element "** et al." is given to Z of a comparison pointer (array of structures), and an unsettled location is changed into "2" from "0." Thereby, an array of structures is set to (2, 1, 2). The increment of the number of front insertion "Y" and virtual ranking number "Z" after such processing and in an array of structures is carried out (step 1320). Thereby, an array of structures is set to (2, 2, 3) (refer to drawing 11 (d)). The virtual ranking number obtained at step 1320 turns into a ranking number of the element (at the above-mentioned example, it is also "****") received at step 1312, and MPU of PMM of the second half transmits the ranking number (the above-mentioned example "3") concerned to PMM of the first half (step 1321). After such processing, the increment of the virtual ranking number is carried out further (step 1322). This is because the ranking number of the following element will become bigger [one] at least than the ranking number given this time.
[0050] PMM of the first half stores the received ranking number as a ranking number of the corresponding element, and, subsequently increments a PUT pointer. Thus, the ranking number of the element in PMM of the first half is decided. In PMM of the first

half, the value MPU of PMM of the first half indicates termination to be for an unsettled element not to already exist (that is, a ranking number is decided about all elements and the element is not arranged in the location of a PUT pointer) is transmitted to PMM of the second half (step 1306 reference). The value which shows termination here is a bigger value than the value which shows the element at the tail end of an array. PMM of the second half answers acceptance of the value which shows the above-mentioned termination, and performs processing of the same processing (steps 1312-1322 of drawing 13) as abbreviation. In the above-mentioned example, since the element contained in the range "1" in spite of acceptance of the value which shows termination does not exist, step 1323 is reached through steps 1315 and 1316, and processing is ended (refer to drawing 12 (b)).
[0051] In PMM of the first half, processing is completed by sending out (step 1316 reference) of the value which shows termination, and decision (it is yes (Yes) at step 1336) of the ranking number of all elements. In the same procedure as the above-mentioned processing, a merge application is performed also between PMM 14-3 and PMM 14-4, and thereby, as shown in drawing 12 (c), the ranking number of each element is decided.
[0052] If the sequence number of each element in two PMM(s) is decided, CPU12 will switch a switch and will connect between two PMM groups which each becomes from two PMM(s). Drawing 14 (a) and drawing 14 (b) are drawings showing an example of connection of two PMM groups in PMM shown in drawing 8 , respectively. In drawing 14 (a), PMM 14-1 and 14-2 constitute the 1st PMM group. CPU12 PMM 14-3 and 14-4 constitute the 2nd PMM group. PMM 14-1 and the output of 14-2, Switches 28 and 30 are controlled so that the input of the PMM group 14-3 is connected, and the output of PMM 14-3 and the input of PMM 14-4 are connected and the output of PMM 14-4, and PMM 14-1 and the input of 14-2 are connected (step 709 reference of drawing 7). Or to be shown in drawing 14 (b), a switch may be controlled so that PMM 14-1 and the output of 14-2 are connected with PMM 14-3 and 14-4.
[0053] Drawing 15 (a) and (b) are drawings which expressed typically drawing 14 R> 4 (a) and (b), respectively. Although it becomes clear behind Data given to PMM 14-1 from PMM 14-4, and 14-2 in drawing 15 (a) (among drawing) Referring to the sign ** is data (among drawing) which show a ranking number and are given to PMM 14-3 from PMM 14-1 and 14-2. The data (refer to sign ** among drawing) which referring to the sign ** shows an element, and are given to PMM 14-4 from PMM 14-3 show the virtual ranking number which an element and PMM 14-3 computed. moreover, drawing 15 (b) -- also setting -- every -- data ** delivered and received between PMM(s) and

** are the same as the thing of drawing 15 (a), and, on the other hand, the data (refer to sign **) transmitted to PMM 14-4 from PMM 14-3 shows the virtual ranking number which PMM 14-3 computed.
[0054] As shown in above-mentioned drawing 12 , explanation is added about the processing (step 709 reference of drawing 7) which determines the merge application in two PMM groups, and the ranking number of an array from the ranking number of the element contained in the part of an array and these in the pair of two PMM(s). in addition, the connection voice of the bus shown in drawing 14 (b) and drawing 15 (b) by the following explanation -- like -- following -- every -- the processing performed in PMM is explained.
[0055] First, each arranges a PUT pointer to an initial position in PMM 14-1 and PMM 14-2 ("the PMM group of the first half" is called hereafter.) (step 1301). In addition, in future processings, a PUT pointer moves in PMM which constitutes the PPM group of the first half according to the element which self holds being sent out. Each of the PMM which it is in the second half on the other hand arranges a comparison pointer to an initial position while initializing the array of structures (step 1302). Subsequently, in each PPM which constitutes the PMM group of the first half, each PMM which constitutes current and the PMM group of the first half grasps of which ranking number the element was sent out. In addition, fundamentally, although the transmitting pointer, receiving pointer, and both sides which use at the time of transmission are used as a PUT pointer in the flow chart, migration of these transmitting pointer and a receiving pointer is performed only with few time difference, although the processing time in the PMM group of the second half is inserted. For example, when a transmitting pointer is made into an increment in a certain PMM so that it may mention later (step 1304 reference), the PMM concerned increments a receiving pointer also in reception (step 1335 reference).
[0056] Each PMM which constitutes the PPM group of the first half judges whether the element concerned is what self is holding based on the ranking number of the element set as the object of processing (step 1302). When judged yes (Yes) at this step 1302, the element to which a PUT pointer points is transmitted to PMM 14-3 and 14-4 through a bus 24 (refer to step 1302 of drawing 13 , and drawing 16 (a)). In the above-mentioned example, the element "****" which is a ranking number "0" is first transmitted to PMM 14-3 from PMM 14-1, and PMM 14-4. By this processing, the location of a PUT pointer moves in PMM 14-1 (step 1304).
[0057] Respectively PMM 14-3 and PMM 14-4 receive an element (step 1312), discover the location which should insert the element (step 1313), and judge whether

the element belonging to the range "1" exists (step 1314). About the above-mentioned element "****", it is judged as a no (No) in step 1314. Thereby, in PMM 14-3, since the virtual ranking number of an element "****" is set to "0", this value is transmitted to PMM 14-4. The virtual ranking number of an element "****" is set to "0" also in PMM 14-4. Then, MPU of PMM 14-4 returns "MAX(0 0) =0" to a front PMM group through a bus as a ranking number of an element "****" (drawing 16 (b) and step 1315 reference). Subsequently, in PMM 14-3 and 14-4, the number of front insertion (Y) and virtual ranking number (Z) in an array of structures increment, respectively (step 1316). Thereby in the above-mentioned example, each array of structures is set to (0, 1, 1), and (0, 1, 1).
[0058] If a ranking number is given from the PMM group of the second half (step 1331), it will judge whether each PMM which constitutes the PMM group of the first half is what self holds [the element under current processing (for example, element "****")] (step 1333). When the ranking number of an element "****" is transmitted, PMM 14-1 judges yes (Yes) at the above-mentioned step 1333, and rewrites the ranking number corresponding to the element of the location to what it was given from the PMM group of the second half (drawing 16 (a) and step 1334 reference). Similarly, the PMM group of the first half transmits the element to which the following ranking number was given to the PMM group of the second half. In the above-mentioned example, from PMM 14-2, an element "****" is transmitted (refer to drawing 17 (a)), and a big thing "MAX(1 1) =1" is transmitted to the PMM group of the first half as a ranking number of the element "****" concerned among each virtual ranking numbers of the PMM group of the second half (refer to drawing 17 R> 7 (b)). Moreover, in PMM 14-3 which constitutes the PMM group of the second half, and 14-4, an array of structures is set to (0, 2, 2), and (0, 2, 2), respectively (refer to drawing 17 (b)).
[0059] Furthermore, the PMM group of the first half transmits the element to which the following ranking number was given to PMM of the second half. In the above-mentioned example, an element "** et al." is transmitted from PMM 14-2 (refer to drawing 18 (a)). In PMM 14-3, an element "** et al." is judged to be back from the element "a shelf" which PMM 14-3 concerned holds (step 1313 reference). therefore, PMM 14-3 -- setting -- the range "1" -- an element -- "-- ** -- obtaining -- " -- since [and] an element "is it a shelf?" belongs -- an element -- "-- ** -- obtaining -- " -- and the number of front insertion "Y (= 2)" is applied to the ranking number of an element "is it a shelf?", respectively. thereby -- an element -- "-- ** -- obtaining -- " -- "0+2=2" and the ranking number of an element "is it a shelf?" are determined for a ranking number as "2+2=4" (step 1317 reference).

Subsequently, MPU of PMM 14-3 gives the ranking number "4" of the element of the tail in the range "1" to the virtual ranking number Z of an array of structures (the current value is (0, 2, 2)) (step 1318 reference), and advances an unsettled location (step 1319 reference). (that is, the value of X is set to "2" from "0") Furthermore, MPU of PMM 14-3 is an array of structures (the current value increments the number of front insertion "Y" and virtual ranking number "Z" of (2, 2, 4) (step 1321 reference).). Thereby, an array of structures is set to (2, 3, 5). The virtual ranking number "Z (= 5)" in PMM 14-3 is transmitted to PMM 14-4 through a bus. After that, MPU of PMM 14-3 increments the virtual ranking number "Z" of an array of structures (step 1322 reference). In the above-mentioned example, an array of structures is set to (2, 3, 6) by giving step 1322.
[0060] the element with which PMM 14-4 concerned, on the other hand, holds an element "** et al." in PMM 14-4 -- "-- passing -- a basis -- " -- it is judged that it is located while "being means" (step 1313 reference). therefore, PMM 14-4 -- setting -- the range "1" -- an element -- "-- passing -- a basis -- " -- since it belongs -- an element -- "-- passing -- a basis -- " -- the number of front insertion "Y (= 2)" adds to a ranking number -- having -- thereby -- an element -- "-- passing -- a basis -- " -- a ranking number is determined as "1+2=3" (step 1317 reference). Subsequently, MPU of PMM 14-4 gives the ranking number "3" of the element of the tail in the range "1" to the virtual ranking number "Z" of an array of structures (the current value is (0, 2, 2)) (step 1318 reference), and advances an unsettled location to it (step 1319 reference). (that is, the value of "X" is set to "1" from "0") Furthermore, MPU of PMM 14-4 increments the number of front insertion "Y" and virtual ranking number "Z" of an array of structures (the current value is (1, 2, 3)) (step 1321 reference). Thereby, an array of structures is set to (1, 3, 4).
[0061] The virtual ranking number to which PMM 14-4 was given from PMM 14-3 next "Z (= 5)", The virtual ranking number "Z (= 4)" which self computed is compared, and "MAX(5 4) =5" which is the value of the bigger one is transmitted to the PMM group of the first half as ranking of the element "** et al." to which it was transmitted (step 321 reference). Thereby, in the PMM group of the first half (setting to PMM 12-2 which sent out the element "** et al." to the detail), it is decided that the ranking number of the element concerned is "5." In addition, also in PMM 14-4, the increment of the virtual ranking number in an array of structures "Z" is carried out after step 1321 (step 1322 reference). An array of structures is set to (1, 3, 5) in the above-mentioned example.
[0062] similarly, an element "****" sends out from the PMM group of the first half --

having (drawing 19 (a)) -- processing in this case is also performed according to drawing 13 . If it explains briefly again, since the element which belongs ahead in the range "1" from the insertion point of an element "****" does not exist in PMM 14-3 which received the element "****", PMM 14-3 transmits the virtual ranking number in the array of structures "Z (= 6)" to PMM-4. Since the element which belongs ahead in the range "1" from the insertion point of an element "****" does not exist in PMM 14-4, The virtual ranking number in the array of structures "Z (= 5)" is compared with the transmitted virtual ranking number "Z (= 6)", and the bigger one (MAX(6 5) =6) of it is returned to the PMM group of the first half as a ranking number of an element "****" (drawing 19 (b) and step 1315 reference). In the PMM group of the first half, PMM 14-1 which sent out the element "****" rewrites the ranking number corresponding to an element "****" for the received ranking number (= 6). In addition, in PMM 14-3, by passing through step 1316, the array of structures is set to (2, 4, 7), and, on the other hand, the array of structures is set to (1, 4, 6) by [the] passing through step 1316 in PMM 14-4.
[0063] Thus, after sending out of all elements is completed in the PMM group of the first half, which PMM which constitutes the PMM group of the first half transmits the value which shows termination to the PMM group of the second half (step 1306 reference). Each PMM which constitutes the PMM group of the second half receives this, and performs processing of step 1312 thru/or step 1323, respectively. In the above-mentioned example, the element "they are means" which has not decided ranking exists in PMM 13-4. For this reason, in PMM 13-4, in step 1314, it is judged yes (Yes), and the number of front insertion "Y" is applied to the ranking number of the element "they are means" belonging to the range "1", and let "3+4=7" and the obtained number "7" be the ranking numbers of an element "they are means." After passing through such processing, in each PMM which constitutes the PMM group of the second half, it is judged yes (Yes) at step 1323, and the processing in the PMM group of the second half is also ended.
[0064] the case where the element under array is stored in PMM beyond it although the element under array was stored in four PMM(s) in the above-mentioned example -- further -- four PMM(s) -- a group, as PMM, each creates the pair of the PMM group which consists of four PMM(s), and should just perform the same processing as abbreviation among these pairs. For example, as shown in drawing 20 , I think that the element under a certain array is stored in 1024 PMM(s). PMM1, PMM2 and PMM3 and PMM4, PMM5 and PMM6, --PMM1023, and PMM1024 are connected first, respectively (refer to the continuous line between PMM(s)). In this case, between

these two PMM(s) The ranking number of an element is decided. Subsequently PMM1 and PMM2 The PMM group of the first half, The pair of the PMM group which makes PMM3 and PMM4 the PMM group of the second half, and PMM5 and PMM6 The PMM group of the first half, The pair of the PMM group which makes PMM7 and PMM8 (not shown) the PMM group of the second half, -- The pair of the PMM group which makes PMM1021 and PMM1022 (not shown) the PMM group of the first half, and makes PMM1023 and PMM1024 the PMM group of the second half is formed. Between each pair is connected (refer to broken line), and the ranking number of an element is decided between two PMM groups which constitute these pairs. The pair of the PMM group which makes hereafter four PMM(s) which follow the PMM group of the first half, and this in four PMM(s) the PMM group of the second half (refer to alternate long and short dash line), Like the pair (refer to dotted line) of the PMM group which makes eight PMM(s) which follow the PMM group of the first half, and this in eight PMM(s) the PMM group of the second half, each carries out sequential formation of the pair of the PMM group which consists of a PMM group which is 2n, and decides the ranking number of an element among these. It becomes possible by deciding the ranking number of an element to decide the ranking numbers of all the elements in 1024 PMM(s) between the pairs of the PMM group which makes eventually 512 PMM(s) which follow the PMM group of the first half, and it in 512 PMM(s) the PMM group of the second half.
[0065] Thus, the pair of the PMM group which each becomes from 2nPMM is formed. By carrying out sequential decision of the ranking number of the element stored in each PMM of the PMM group which constitutes a pair, (Step 709 of drawing 7 , 710 reference), If the ranking number of all elements is decided eventually (yes (Yes), processing which carries out the reconstititution of the array according to the above-mentioned ranking numbering is performed at step 710 when required (step 711).) Although this processing is not indispensable, it becomes possible to realize more information processing performed behind at a high speed by generating an array by which the element is arranged according to the ranking number.
[0066] In a detail, CPU12 controls switches 28 and 30 first more to connect with the bus by which the input and output of each PMM are. Drawing 21 is drawing showing connection between these typically, when PMM is four. Subsequently, MPU of PMM 14-1 to 14-4 emits an element and a ranking number on a bus according to the settled ranking number. Each MPU acts as the monitor of the element emitted on a bus, and its ranking number, incorporates the element which has the same ranking number as the suffix (record number) of the element alloted with the RAM core of self from the

first, and stores it in the predetermined field of a RAM core. For example, what is necessary is to incorporate the element to which a ranking number ″0″ and ″1″ were given, and just to memorize these in PMM which had memorized the suffix (record number) ″0″ and the element of ″1″ to the RAM core of self from the first (for example, 14 to PMM1 reference of drawing 10). If it does in this way, it will become possible to allot each array actually sorted in PMM. In addition, also in case the sorted array is formed in this way, MPU of PMM creates a required space ID managed table.
[0067] Or other PMM(s) (PMM14-5-PMM 14-8) for alloting the sorted array, as shown in drawing 22 are prepared. From PMM14-1-PMM 14-4, each of other PMM groups may act as the monitor of the element by which a sequential output is carried out, and its ranking number, may incorporate the element which self should incorporate according to a ranking number, and may memorize to each RAM core of PMM. For example, 1024 PMM(s) are prepared using above-mentioned this invention, and when [each] about 1 million data (element) are stored in PMM and these data are sorted, it is thought that a sort is completed by the following time amount. All PMM(s) that as for the bus which connects between each PMM all PMM(s) operate here in juxtaposition (that is, PMM which is not performing processing does not exist), and relate to it during processing possible [6.4GB/second of data transmission] assume that it can said-operate that it is simultaneous and cooperatively. Moreover, I think that the sort of each about 1 million data (element) in PMM is completed in 2.5 seconds. In this case, in order to sort about 1 billion elements in 1024 PMM(s), it turns out that it needs only about abbreviation 4 second.
[0068] According to the gestalt of this operation, each PMM is divided into the pair of two PMM(s) in first stage, subsequently, it divides into the pair of the PMM group by which each group is constituted from 2nPMM, and the ranking number is made to decide between each pair one by one. Moreover, decision of the ranking number in each pair can perform in juxtaposition by adjusting the bus used in each pair using a switch etc. Furthermore, the ranking number in each pair can be decided by repeating the procedure of transmitting the ranking which transmitted the element from the PMM group of the first half to the PMM group of the second half, was made deciding it according to the value in the array of structures in the PMM group of the second half, and was decided to the PMM group of the first half. Therefore, while being able to perform processing very in juxtaposition, without PMM (″it playing″) which is not performing processing arising, the amount of data transfer using a bus is reducible. [so-called] This becomes possible to make a sort rate into a high speed remarkably.
[0069] In addition, in the gestalt of implementation of the above 1st, as shown in

drawing 14 (b) and drawing 15 (b), PMM is connected, and although sorting application is realized to ****** which gives a ranking number to each element among these, as shown in drawing 14 (a) and drawing 15 R> 5 (a), PMM may be connected. In this case, if processing (step 1312 - step 1323) of the PMM group of the second half in drawing 13 is not performed in juxtaposition but a virtual sequential number is obtained in a certain PMM, the element used as a processing object and the virtual ranking number concerned will be transmitted to adjoining PMM, and processing of steps 1312-1323 will be performed in the PMM concerned. Therefore, if the number of PMM which constitutes the PMM group of the second half increases, delay of processing may be caused so much.
[0070] Explanation is added per gestalt of sorting application (gestalt of the 2nd operation)] besides [, next operation of the 2nd of this invention. In the gestalt of implementation of the above 1st, all elements (element in the PMM group of the first half) are transmitted to the PMM group of the second half. However, many duplication values may appear as an array becomes huge. By the technique concerning the gestalt of implementation of the above 1st, the element which takes the same value is repeatedly sent out on a bus. It is possible that it is useless to repeat and send out the element of the same value depending on the case. Then, in the gestalt of the 2nd operation, it has prevented repeating the overlapping element and sending out on a bus by counting the number of the element in a PMM group beforehand, and sending out the number to the PMM group of the second half with an element.
[0071] For example, it considers sorting application being completed in a pair of each of four PMM(s), connecting these pairs, and performing sorting application in eight PMM(s). In this case, as shown in drawing 23 , it is desirable that transfer of the data between PMM(s) can be performed using other buses (the bus 2304, 2305 reference which are located in a PMM upside in drawing 23) besides [which performs merge and sorting application of eight PMM(s)] a bus (the bus located in the PMM bottom in drawing 23 , for example, 2301 to sign 2303 reference). In a connection mode as shown in drawing 2323 , explanation is added per [which computes the number of duplications of the value in PMM14-1-PMM 14-4 (PPM / "PPM 14-1" / thru/or "PMM 14-4" are hereafter called for convenience PMM / "PMM1" / thru/or "PMM4", respectively.)] processing. The bus (sign 2304 reference) connected with the input/output terminal (I/O) of PMM1-PMM4 is called the 1st bus here, and the bus (sign 2305 reference) connected with other input/output terminals (I/O) of PM1-PMM4 is called the 2nd bus. The 1st bus is used for information interchange of the PMM group which consists of PMM1-PMM4, and the 2nd bus is used in order to

give a value and its number of duplications to each PMM.
[0072] In addition, in the following explanation, as shown in drawing 25 , although the ranking number was given to each element, the number of duplications is computed in the array in PMM1-PMM4. That is, if it computes only in the PMM group of the first half, it is sufficient for the number of duplications. Drawing 24 is a flow chart which shows the processing for computing the number of duplications in a PMM group. Each of PMM1-PMM4 performs processing of various initialization first (step 2401). The ranking number counter which shows the ranking number of each value (element) applied to processing in PMM here, The same value number counter with which a certain value (element) shows whether which overlaps and it exists, And a value-preserving register is prepared last time holding the value (element) which became a processing object in the last processing, and initial value "0" is given to the value of a ranking number counter and the same value number counter (refer to drawing 25). In addition, no values are held last time in first stage at a value-preserving register.
[0073] subsequently, every -- PMM specifies the ranking number of the element used as a processing object with reference to a ranking number counter, and judges whether the element to which the ranking number concerned was given is what self holds (step 2403). In the above-mentioned example, in first stage, since the counter value of a ranking number counter is "0", PMM3 judges that the element which self holds is a processing object (being step 2403 yes (Yes)). In addition, the following steps 2404-2405 are disregarded by the first processing (namely, processing about the element of a ranking number "0"). The number of the existence in self-PMM the element as the element (in this case, "****") to which the ranking number "0" was given with same PMM3 indicates it to be how many it has an element "****" and this element into the 1st bus by judging how many it exists in (that is, how many is PMM3 holding the element "****"?) is sent out (step 2406). In other PMM(s) (PMM1, PMM2, and PMM4), since it is judged as a no (No) at step 2403, it progresses to step 2407.
[0074] Each PMM receives the data given through the 1st bus, and applies the number of the existence in PMM to the counter value of a ranking number counter based on the number of the existence in self-PMM in data (step 2408). The counter value of a ranking number counter is set to "0+1=1" in the above-mentioned example. Subsequently, it is judged whether the given element differs from the thing of a value-preserving register last time (step 2409), and when both sides are the same, the number of the existence in self-PMM is applied to the counter value of the same value number counter (step 2410), and when it is the new value, on the other hand,

exchange processing mentioned later is performed (step 2411). In addition, in first-time processing, since the value is not held at all last time at a value-preserving register, while decision of the above-mentioned step 2409 is omitted and an element is held in a value-preserving register last time, count-up of the same value number counter is performed. Therefore, in the above-mentioned example, each PMM sets the same value number counter to "0+1=1" while memorizing the received element "****" to a value-preserving register last time (refer to the drawing 2626). After processing of such steps 2401-2411 is repeated and the processing about the last element is completed, in step 2401, it is judged yes (Yes), and progresses to step 2412.
[0075] if processing of the first steps 2401-2411 is completed in the above-mentioned example -- every -- PMM checks that a counter value is "1" with reference to the counter value of a ranking number counter. Thereby, it turns out that PMM4 is holding the element of a ranking number "1." Moreover, since PMM4 compares last time the element "****" to which the value (element "****") and ranking number "1" of a value-preserving register were given (step 2404) and there is no change in a value, an element "****" and the number of the existence in self-PMM "1" are sent out to the 1st bus (step 2405). Since the value with which each PMM which received data through the 1st bus counted up the ranking number counter (1+1=2) (step 2408), and was remembered to be by the value-preserving register last time as shown in drawing 27 , and the received element are the same, the same value number counter is counted up (step 2410). (1+1=2)
[0076] In each PMM, processing about the element of a ranking number "2" is performed by next. By processing of the element of a ranking number "2", since PMM1 holds the element, PMM1 compares an element "******" with the element "****" memorized by the value-preserving register last time. Here, since a value has change (it is yes (Yes) at step 2404), PMM1 sends out the content (element "****") of the value-preserving register, and the value "2" of the same value number counter to the 2nd bus last time (step 2405). The content and counter value of this register are given to each PMM. When the number of duplications of a certain element (in this case, element "****") is computed so that it may mention later, sorting application (refer to drawing 31) about the element concerned may be performed. Therefore, what is necessary is just to hold an element and its number of duplications in each PMM, until the sorting application about the element concerned is completed. Moreover, an element "******" and the number of the existence in self-PMM "1" are given to the 1st bus (step 2406).
[0077] Each PMM counts up a ranking number counter based on the data given

through the 1st bus (step 2408). (2+1=3) since [moreover,] the elements "******" delivered the value "****" of a value-preserving register differ last time (it is yes (Yes) at step 2409) -- every -- PMM -- last time -- the value of a value-preserving register -- rewriting (it updating) -- it transposes to the number of the existence in self-PMM to which the value of the same value number counter was given through the 1st bus (refer to step 2411 and drawing 28 (a)).
[0078] Same processing is performed also with the element of other ranking numbers. For example, about a ranking number "3", PMM3 sends out an element "******" to the 1st bus according to steps 2404 and 2406, and each PMM counts up each counter according to steps 2407, 2408, 2409, and 2410 (refer to drawing 28 (b)). Moreover, while PMM1 sends out the counter value "2" of an element "******" and the same value number counter to the 2nd bus about a ranking number "4" according to steps 2404, 2405, and 2406 an element -- "-- passing -- a basis -- " -- the 1st bus -- sending out -- and every -- in order of steps 2407, 2408, 2409, and 2411, PMM updates a register, while counting up each counter (refer to drawing 29 (a)).
[0079] the element with which self holds PMM2 in the processing about a ranking number "5" -- "-- passing -- a basis -- " -- the 1st bus since there are two -- an element -- "-- passing -- a basis -- " -- and the number of the existence in self-PMM "2" is sent out. Therefore, in each PMM, "2" is added to each counter value of a ranking number counter and the same value number counter (refer to drawing 29 R> 9 (b)). Moreover, by this processing, since the counter value of a ranking number counter changes to "7" from "5", please care about that the ranking number of the element used as the following processing object is set to "7" instead of "6." After the processing (refer to drawing 30 (a)) about the last element to which the ranking number "7" was given is completed, it is judged yes (Yes) at step 2401. then, top PMM (the above-mentioned example PMM1) -- the 2nd bus -- an element -- "-- passing -- a basis -- " -- and the counter value "4" of the same value number counter is sent out (step 2413), and the data in which it is shown subsequently to the 2nd bus that processing was completed are sent out (step 2414). The number of existence which shows each element and its number is given to each PMM through the 2nd bus, and this is used for sorting application. In addition, what is necessary is not to be limited to this and just to define PMM which outputs beforehand data in which termination is shown, such as the last element, although top PMM consisted of above-mentioned examples so that steps 2413 and 2414 might be performed. As mentioned above, in case a PMM group is merged with other PMM groups and these elements are sorted by obtaining the number of existence of each element in a certain

PMM group, it becomes unnecessary to send the duplicate element.
[0080] Drawing 31 is a flow chart which shows the sorting application which eliminated sending out of the duplicate element. Drawing 31 is the same as that of processing of drawing 13 except for a part, and the thing with the double figures same tail serves as processing which carries out an abbreviation response. Moreover, in drawing 31 , it is shown that the processing which attached the enclosure of a duplex is the processing added newly or different processing that to which drawing 13 corresponds, and a little. In this processing, PMM which holds the element (namely, element directed by the sending-out pointer) used as a processing object sends out the number of duplications of the element concerned in the PMM group of the first half (the number of existence) "N" to the PMM group of the second half with that element in the PMM group of the first half (step 3103, 3103-2 reference). For example, in the example shown in drawing 25 thru/or drawing 30 , from the PMM group of the first half which consists of PMM1-PMM4, when an element "****" is sent out to the PMM group of the second half, the number of duplications "2" of the element "****" in the PMM group of the first half besides an element "****" is transmitted. Moreover, in sending-out processing of the PMM group of the first half, only the number of the elements concerned which self grasps moves [PMM / which outputted an element and its number of duplications] a sending-out pointer after the output (step 3104). For example, as shown in drawing 28 (a), the number of duplications of an element "****" is "2", and these [one / every] are grasped in PMM3 and PMM4. Therefore, in PMM3 and PMM4, one location of a sending-out pointer moves caudad, respectively. in addition, every -- total of the movement magnitude of the sending-out pointer in PMM becomes equal to the number of duplications of the element concerned "N."
[0081] In each PMM which constitutes the PMM group of the second half which, on the other hand, received an element and its number of duplications, as shown in step 3116 and step 3120 of drawing 31 R> 1, the number of duplications "N" is applied to the number of front insertion, and a virtual ranking number, respectively. This supports that the element (ranking is small) ahead located from itself exists only in "N."
[0082] Furthermore, in the reception of PMM which constitutes the PMM group of the first half, the difference "M" of the received ranking number and the ranking number at the time of sending out of the data for a comparison is computed based on the element (data for a comparison) sent out in the sending-out processing by PMM of the first half, and the ranking number sent out in processing by PMM of the second half (step 3132-2). This difference "M" shows the number of the elements (that is, the ranking number smaller than the element concerned was attached) located ahead of

the element used as the object for a comparison in the PMM group of the second half. Therefore, the same element as the element which serves as the object for a comparison concerned among the elements with which self holds each PMM which constitutes the PMM group of the second half is specified (step 3132-3), and in existing, it adds "M" to the ranking number of these elements, respectively (step 3134). After step 3134, PMM moves [number / of the elements concerned] a receiving pointer (step 3135). This processing is the same as that of step 3104 and abbreviation.
[0083] Next, explanation is added per [of calculation of the number of duplications shown in drawing 24 , and the sorting application (a "sort body" is called by the case.) shown in drawing 31 R> 1] parallelism. As shown in drawing 23 , in the gestalt of this operation, buses 2301 and 2302 and 2303 grades are used for the communication link between PMM(s) which use buses 2304 and 2305 and start activation of a sort body in the communication link between PMM(s) concerning the count of the number of duplications. Then, if parallel processing is possible in PMM, the count and sort body of the number of duplications can be arranged in parallel and performed. In this case, it is if calculation of the number of duplications about a certain element is completed in the PMM group of the first half (for example, as shown in drawing 28 (a)). An element "****" and its number of duplications "2" are sent out to the 2nd bus, and if received by PMM (PMM1-PMM4) which constitutes the PMM group of the first half, processing as shown in drawing 3131 can perform about the element with which the number of duplications was computed. That is, calculation of the number of duplications of a certain element can be answered, and processing of step 3102 about the element concerned - step 3104, processing of step 3112 - step 3122, and processing of step 3132 - step 3135 can be performed. Moreover, although the element about a certain element, its number of duplications, etc. are related with the element concerned among the processings shown in above-mentioned drawing 31 , they can be deleted with termination. Therefore, in each of PMM which constitutes the PMM group of the first half, it is not necessary to hold all (for the number of different elements to follow the amount on increasing, and it to become large) of an element and the data about the number of duplications.
[0084] Thus, in the gestalt of the 2nd operation, in the PMM group of the first half, the number of duplications was computed and an element and its number of duplications are sent out to the PMM group of the second half. It becomes unnecessary therefore, for the PMM group of the first half to overlap and send the same element to the PMM group of the second half. When many same elements overlap especially (for example,

that an element indicates man and woman's classification to be, the thing which shows age), it becomes possible to decrease the count of processing of a sort body, and sorting application can be realized more at a high speed.
[0085] Explanation is added per gestalt of [compile processing (gestalt of the 3rd operation)], next implementation of the 3rd of this invention. With the gestalt of the 3rd operation, the pointer array for specifying a record to a record, the value list which has arranged each element without duplication, and a value list based on the array which consists of an element arranged in each PMM is created. This processing is called compile in this description. For example, what is necessary is just to connect PMM, as shown in drawing 32 when a certain array element is alloted by four PMM(s) (PMM1-PMM4). drawing 32 -- being shown -- as -- PMM -- one - PMM -- four -- an input/output terminal (I/O) -- the -- one -- a bus (sign 3201 reference) -- connecting -- having -- the -- on the other hand -- PM -- one - PMM -- four -- an output terminal -- (-- O --) -- and -- others -- PMM"k -- " -- an input terminal -- (-- I --) -- the -- two -- a bus (sign 3202 reference) -- connecting -- having -- **** .
[0086] The 1st bus is used for information interchange of the PMM group which consists of PMM1-PMM4, and the 2nd bus is used in order to give an element and its number of duplications to other PMM"k." In the gestalt of this operation, a value list, the number array of existence, etc. are formed in other PMM"k" based on the above-mentioned element and its number of duplications. In addition, although this PMM"k" may be PMM(s) other than PMM1 - PMM4, of course, it may be in any of PMM1-PMM4. Drawing 33 is a flow chart which shows the compile processing concerning the gestalt of this operation. In addition, in order to give explanation easy, as shown in drawing 34 (a), the element is alloted by PMM1-PMM4, and I think to them that processing which attaches a ranking number among these has already been performed. First, the ranking number counter which shows the ranking number of each value (element) applied to processing in PMM, The value number counter which shows the ranking number of the value concerned after processing (element), the same value number counter with which the element concerned shows whether which overlaps and it exists, And a value-preserving register is prepared last time holding the value (element) which became a processing object in the last processing, and initial value "0" is given to each counter (refer to step 3301 and drawing 34 (a)). In addition, a value is not held last time in first stage at a value-preserving register.
[0087] Processing of step 3302 of the following and drawing 33 - step 3306 is the same as that of steps 24021-2406 of drawing 24 , and abbreviation. That is, it judges

whether each element to which PMM specified the ranking number with reference to the ranking number counter so that it might become a processing object, and the ranking number concerned was given is what self holds (step 3303). In the state of drawing 34 , since the counter value of a ranking number counter is "0", PMM3 creates the number of the existence in self-PMM (in this case, "1") which shows how many PMM3 concerned holds the element "****" to which the ranking number "0" was given to the 2nd bus (step 3306 reaching (referring to drawing 34 (b)).). Subsequently, when the element memorized by the value-preserving register last time is compared with the element emitted to the 1st bus and these are different, PMM3 assigns the counter value of a value number counter to the ranking number, as sent out to the 1st bus (step 3307). In addition, in the state of drawing 3434 , since the counter value of a value number counter is initial value "0", the ranking number concerning an element "****" does not change (refer to drawing 34 (b)).
[0088] Subsequently, each data given through the 1st bus in PMM is received (step 3308). Processing of steps 3308-3311 is the same as that of processing of steps 2408-2401 in drawing 24 , and abbreviation. namely, every -- PMM applies the number of the existence in PMM of the data given to the counter value of a ranking number counter, and further, when an element is not new among the given data (step 3310 no (No)), it applies the number of the existence in PMM to the counter value of the same value number counter (refer to step 3311 and drawing 34 (b)). As shown in drawing 34 , after the processing about the element "****" to which the ranking number "0" was given is completed, processing about the element to which the ranking number "1" was given is performed similarly (refer to drawing 35 (a)).
[0089] Furthermore, processing about the element to which the ranking number "2" was given is performed. Here, PMM1 compares the element "****" memorized by the value-preserving register last time with the element "******" to which the ranking number "2" was given. Here, since these are different (it is yes (Yes) at step 3304), PMM1 sends out the element memorized by the value-preserving register last time and the counter value of the same value number counter to the 2nd bus (step 305). Subsequently, PMM1 sends out the number of the existence in self-PMM "1" which is the number of the element "******" of a processing object, and the elements "******" which PMM1 holds to the 1st bus (step 3306). After that, PMM1 compares the element memorized by the value-preserving register last time with the element emitted to the 1st bus. Since these are different in emitting an element "******", the value (0+1=1) which added "1" at the counter value of a value number counter is assigned to the ranking number of an element "******." The number of the existence

in self-PMM in the data which received each data with which PMM was given through the 1st bus (step 3308), and received it in the counter value of a ranking number counter is added (refer to step (2+1=3) 3309 and drawing 35 (b)). since the element "****" of a value-preserving register differs from the given element "******" last time when an element "******" is given (it is yes (Yes) at step 3310) -- every -- PMM performs new price registration processing (step 3312). In this processing, the increment (0+1=1) of the counter value of a value number counter is carried out, and the counter value of the same value number counter is changed into the number of the existence in self-PMM in the received data "1", and the content of the value-preserving register is rewritten by the element "******" last time (refer to drawing 35 (b)).
[0090] Same processing is performed also about the element "******" of a ranking number "3." For example, PMM3 sends out an element "******" and the number of the existence in self-PMM "1" to the 1st bus (step 3306 reference), and assigns the counter value "1" of a value number counter to the ranking number of the element "******" concerned (refer to step 3307 and drawing 36 (a)). moreover, every -- the number of the existence in self-PMM "1" which received PMM to the counter value of a ranking number counter -- adding (step 3309 reference) -- the number of the existence in self-PMM "1" is applied to the counter value of the same value number counter (refer to step 3311 and drawing 36 (a)). furthermore, the element of a ranking number "4" -- "-- passing -- a basis -- ", as it is also related and is shown in drawing 3636 (b) PMM1 sends out the counter value "2" of an element "******" and the same value number counter to the 2nd bus (step 3305). the 1st bus -- an element -- "-- passing -- a basis -- " -- and the number of the existence in self-PMM "1" -- sending out (step 3306) -- and an element -- "-- passing -- a basis -- " -- the value (1+1=2) which added "1" at the value number counter is assigned to a ranking number. Also in one of these, and each PMM, count-up of a ranking number counter and new price registration processing are performed (refer to steps 3309 and 3312 and drawing 36 (b)). Same processing is performed also with the element of other ranking numbers. The processing about each element is shown in drawing 37 (a), (b), and drawing 38 . in addition, the drawing 3838 -- being related -- PMM1 -- the last element -- "-- passing -- a basis -- " -- and the data in which the number of existence of the element is sent out to the 1st bus, and termination is shown are outputted to the 2nd bus (step 3315 reference).
[0091] The input of PMM"k" is connected to the 2nd bus as mentioned above. Therefore, an element without duplication and the counter value of the value number

counter about this are given to the 2nd bus. Therefore, PMM"k" receives these, and it carries out sequential arrangement of the counted value of the received value number counter at the number array of existence while it carries out sequential arrangement of the received element at a value list. Drawing 39 (a) is drawing showing the value list and the number array of existence which were created in PMM "k." These are sent out at step 3305 or step 3314 (refer to drawing 35 (b), drawing 36 R> 6 (b), and drawing 38), and are transmitted to PMM"k." As shown in drawing 39 (a), the number of existence (namely, the number of duplications) which shows how many it is arranged without duplication of an element at a value list, and each element exists is arranged at the number array of existence.
[0092] Furthermore, in PMM1-PMM4, the pointer array to the value list to which a record and the ranking number attached so that there might be no duplication in each element are made to correspond can be created. That is, if the array to which the record and the ranking number given to the element corresponding to the record concerned were made to correspond is created, this can consider as the pointer array to a value list (refer to drawing 39 (b)). In drawing 39 (b), the ranking number "2" of a corresponding element serves as pointer value in the pointer array to a value list about a record "0." This shows what a value which is a storing location number "2" should be directed for in the value list (refer to drawing 39 (a)). That is, the value list stored in PMM"k" can be directed with the pointer value of the pointer array to a value list, and this becomes possible to specify an element from a record.
[0093] Thus, according to the gestalt of this operation, it is carrying out swing direct [of the ranking] so that the array element alloted by PMM is sorted, a ranking number may be attached and the same ranking may be given to the same element. An element is matched with ranking without the newly obtained duplication, and is stored in a value list. the element under array in which the ranking concerned was alloted as a pointer array to a value list -- a response -- the price -- ******** . Therefore, based on a record, it becomes possible through the pointer value of a pointer array to specify the element under value list.
[0094] Explanation is added per gestalt of [share-izing (gestalt of the 4th operation) of a value list], next implementation of the 4th of this invention. Two arrays are share-ized in the gestalt of the 4th operation (join). As this premise, the pointer array to the value list by compile processing and a value list is created. Moreover, Space ID was given to the pointer array to a value list and a value list, and each PMM grasps in it the various information about the array which self allots by the space ID concerned etc.

[0095] Drawing 40 is a flow chart which shows the share-ized processing concerning the gestalt of the 4th operation. In order to give explanation easy, as shown in drawing 41 (a), the array (sign 4100 reference) which consists of an element corresponding to a record as original data thinks that it is alloted by a certain PMM group. The block (a "information block" is called hereafter.) which becomes the PMM group which consists of PMM1 and PMM2 from a pointer array (sign 4101 reference) and a value list (sign 4102 reference) by compile processing about this record group is formed. I think that the information block to which other arrays (sign 4110 reference) which consist of the element corresponding to a record as original data are alloted by other PMM groups, and become the PMM group which consists of a PMM group 3 and a PMM group 4 from a pointer array (sign 4111 reference) and a value list (sign 4112 reference) by compile processing on the other hand is formed.
[0096] The instruction which directs the join of a value list is transmitted to each PMM from CPU12 with the space ID of the array which shows two value lists. every -- a thing [as / whose array which should be carried out a join among PMM(s) is the value list which self holds, or its part] (that is, the above-mentioned example PMM1-PMM4) specifies the value list set as the object of a join based on Space ID (refer to step 4001 and drawing 42 (a)). Subsequently, PMM1-PMM4 are in the condition which joined two value lists, sort these and give a ranking number to each element (step 4002). The sorting application which starts the gestalt of the 1st operation for this sorting application can be used. In the above-mentioned example, the 1st PMM group which consists of PMM1 and PMM2, and the 2nd PMM group which consists of PMM3 and PMM4 are alike first, respectively, it sets, and processing of ranking numbering of an element is performed. subsequently A ranking number is given to the element in two PMM groups by making the 1st PMM group into the PMM group of the first half, and making the 2nd PMM group into the PMM group of the second half. Drawing 42 (b) is drawing showing the condition that did in this way and the ranking number was given to the element.
[0097] Compile processing is performed between PMM(s) which allot the value list used as a processing object after that, and the number array of existence which was communalized by any [other PMM(s) or] of PMM1-PMM4 they are by this and which was communalized [which were communalized and was value-listed] is generated (step 4003). That is, a new value list which the element of the merged value list does not overlap by compile processing, and the number array of existence in which each element stored the number of existence which shows which overlaps and exists are acquired (refer to drawing 42 (c)). The new pointer array for directing the new value

list (namely, value list obtained by compile processing) by which the join was carried out after such processing is searched for. This is realized by creating a new pointer array which stores in the location of the pointer value under pointer array concerned, and a corresponding location the ranking number which the pointer value under pointer array in the information block before share-izing shows, and to which the ranking number array acquired by compile processing corresponds. Dealing with the new ranking number (referring to drawing 42 (c)) given to each element could understand the value under above-mentioned ranking number array.
[0098] As shown in drawing 43 (a), for example, since the value (ranking number) under ranking number array of the location which the 1st pointer value under pointer array "1" shows is "2", the pointer value of the location where it corresponds under pointer array after being share-ized is set to "2." Moreover, since the value (ranking number) under ranking number array of the location which the 2nd pointer value "2" shows is "3", the pointer value of the location where it corresponds under share-ized pointer array is set to "3." Thus, it becomes possible to acquire the pointer array about the value list by which the join was carried out (drawing 43 (a) and drawing 43 (b)).
[0099] Probably, it will be clear that a value (element) can be specified from a record with such a new pointer array and the value list by which the join was carried out. As shown in drawing 44 , the pointer value of the location where a record corresponds during the pointer array to the newly obtained value list is specified, and the element under value list in the location which the value shows [the pointer value concerned] is specified. Here, that the same element as the element of the original data is specified could understand, in spite of carrying out the join of the two value lists.
[0100] Thus, according to the gestalt of the 4th operation, the ranking number array of the value list by which the join was carried out, and each value list is acquired by merging two or more value lists and combining sorting application and compile processing about the element of the merged value list. The value (ranking number) of a ranking number array is specified according to the pointer array for specifying a value list from a record, and the new pointer array for specifying the value list by which the join was carried out based on the record can be acquired by storing the ranking number concerned in the location corresponding to a record. Therefore, it becomes possible to carry out the join of two or more value lists with the time amount of the sorting application mentioned above, and time amount extent of compile processing, and it becomes possible to raise processing speed remarkably.
[0101] Various modification is possible for this invention within the limits of invention

indicated by the claim, without being limited to the gestalt of the above operation, and it cannot be overemphasized that it is that by which they are also included within the limits of this invention. For example, in the gestalt of said operation, although it has applied to the computer system, it is not limited to this, and this invention can also be applied to a computer board connectable with a personal computer etc. In this case, in drawing 1 , CPU12, the memory unit 14, and bus 24 grade are carried on a board, and this constitutes the information processing unit in this invention.
[0102] Moreover, the number of the groups of the bus which connects between CPU12 and memory modules 14 and/or between memory modules 14 is not limited to the gestalt of said operation, and can be suitably determined in consideration of the magnitude of the circuit board in which a computer system is carried, the number of bits of each bus, etc. Moreover, in the gestalt of said operation, it is between the switch 28 for specifying connection with I/O and bus of a memory module, and CPU and a memory module, between memory modules, or between I/O of a memory module, and the switch 30 from which a bus can be cut is formed. forming switches 29 and 30 -- for example, while using a certain bus (24 to bus 4 reference of drawing 1) for data transfer with the CPU module 12 and a memory module 14-1, it can be simultaneously used for the data transfer between a memory module 14-2 and a memory module 14-3 (in this case, what is necessary is just to turn OFF a switch 30-5). Therefore, it is possible to use a bus for validity more. However, when the group of a bus can enlarge a number enough, or when there are comparatively few memory modules, it is not necessary to necessarily prepare a switch 29 or 30.
[0103] Moreover, in this description, although it indicated that the instruction from CPU12 was given through the control signal line 25, it cannot be overemphasized that various control signals, such as a clock besides an instruction, for each memory module to synchronize and operate are given through the control signal line 25, and the predetermined signal (for example, an error signal and the signal which shows data acceptance) from each memory module to CPU12 is given.
[0104] Furthermore, in the gestalt of said operation, although the various connection between PMM(s) was illustrated, selection of the bus used for the connection and transmission and reception between PMM(s) is not limited to what is shown in the gestalt of the above-mentioned implementation.
[0105] Moreover, in the gestalt of said 3rd operation, as shown in drawing 32 , the 1st bus (sign 3201) is used. It is not what is limited to this although the number of existence of an element and the element concerned (the number of duplications) is communicating the communication link between each PMM using nothing and 2nd

refer to the bus (sign 3202). For example, as shown in drawing 45 , PMM″k″ which generates the value list which is the array of an element without duplication, and its number array of existence acts as the monitor of the 1st bus 4501. Based on the element and the number array of existence which appear on the 1st bus 4501, predetermined processing (for example, maintenance/renewal of count-up of a counter or the content of the register performed in PMM1-PMM4) may be performed, and a value list and the number array of existence may be created.
[0106] Furthermore, in this description, even if the function of one means is realized by two or more physical means, the function of two or more means may be realized by one physical means.

[Translation done.]

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] Drawing 1 is a block diagram which shows the configuration of the computer system concerning the gestalt of operation of this invention.
[Drawing 2] Drawing 2 is a block diagram which shows the outline of the memory module concerning the gestalt of this operation.
[Drawing 3] Drawing 3 is drawing for explaining the pipeline processing between the memory modules concerning the gestalt of this operation.
[Drawing 4] Drawing 4 is drawing for explaining the structure of a memory module 14 under the multi-space memory concerning the gestalt of this operation.
[Drawing 5] Drawing 5 is drawing for explaining access to the memory module in the gestalt of this operation.
[Drawing 6] Drawing 6 is drawing showing an example of an array which performs sorting application concerning the gestalt of the 1st operation.
[Drawing 7] Drawing 7 is a flow chart which shows the procedure of the sorting application concerning the gestalt of the 1st operation.
[Drawing 8] Drawing 8 is a block diagram which shows connection between the memory modules at the time of carrying out sorting application concerning the gestalt of the 1st operation.

[Drawing 9] Drawing 9 is drawing showing typically connection between the memory modules shown in drawing 8 .
[Drawing 10] Drawing 10 is drawing showing numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 11] Drawing 11 is drawing showing ranking numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 12] Drawing 12 is drawing showing numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 13] Drawing 13 is a flow chart which shows the ranking number processing between the pairs of the memory module concerning the gestalt of the 1st operation.
[Drawing 14] Drawing 14 is a block diagram about the memory module shown in drawing 8 which shows the example of connection of two memory module groups.
[Drawing 15] Drawing 15 is drawing showing typically the example of connection shown in drawing 14 .
[Drawing 16] Drawing 16 is drawing showing ranking numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 17] Drawing 17 is drawing showing ranking numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 18] Drawing 18 is drawing showing ranking numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 19] Drawing 19 is drawing showing ranking numbering to the element under array in the sorting application concerning the gestalt of the 1st operation.
[Drawing 20] Drawing 20 is drawing for explaining the gestalt of the 1st operation, or the combination of the memory module in the sorting application to cut.
[Drawing 21] Drawing 21 is drawing showing the example of connection of the memory module in the case of generating a new array according to the ranking number obtained as a result of the sorting application concerning the gestalt of the 1st operation.
[Drawing 22] Drawing 22 is drawing showing other examples of connection of the memory module in the case of generating a new array according to the ranking number obtained as a result of the sorting application concerning the gestalt of the 1st operation.
[Drawing 23] Drawing 23 is drawing showing typically the example of connection of the memory module in the sorting application concerning the gestalt of the 2nd operation.
[Drawing 24] Drawing 24 is drawing for explaining the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd

operation.
[Drawing 25] Drawing 25 is drawing for explaining the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd operation.
[Drawing 26] Drawing 26 is a flow chart which shows the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd operation.
[Drawing 27] Drawing 27 is drawing for explaining the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd operation.
[Drawing 28] Drawing 28 is drawing for explaining the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd operation.
[Drawing 29] Drawing 29 is drawing for explaining the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd operation.
[Drawing 30] Drawing 30 is drawing for explaining the processing for computing the number of duplications in the memory module group concerning the gestalt of the 2nd operation.
[Drawing 31] Drawing 31 is a flow chart which shows the sorting application which eliminated sending out of the duplicate element.
[Drawing 32] Drawing 32 is drawing showing typically the example of connection of the memory module in the compile processing concerning the gestalt of operation of the 3rd of this invention.
[Drawing 33] Drawing 33 is a flow chart which shows the compile processing concerning the gestalt of the 3rd operation.
[Drawing 34] Drawing 34 is drawing for explaining the compile processing in the memory module group concerning the gestalt of the 3rd operation.
[Drawing 35] Drawing 35 is drawing for explaining the compile processing in the memory module group concerning the gestalt of the 3rd operation.
[Drawing 36] Drawing 36 is drawing for explaining the compile processing in the memory module group concerning the gestalt of the 3rd operation.
[Drawing 37] Drawing 37 is drawing for explaining the compile processing in the memory module group concerning the gestalt of the 3rd operation.
[Drawing 38] Drawing 38 is drawing for explaining the compile processing in the memory module group concerning the gestalt of the 3rd operation.

[Drawing 39] Drawing 39 is drawing for explaining the compile processing in the memory module group concerning the gestalt of the 3rd operation.
[Drawing 40] Drawing 40 is a flow chart which shows the share-ized processing concerning the gestalt of operation of the 4th of this invention.
[Drawing 41] Drawing 41 is drawing for explaining the join processing in the memory module group concerning the gestalt of the 4th operation.
[Drawing 42] Drawing 42 is drawing for explaining the join processing in the memory module group concerning the gestalt of the 4th operation.
[Drawing 43] Drawing 43 is drawing for explaining the join processing in the memory module group concerning the gestalt of the 4th operation.
[Drawing 44] Drawing 44 is drawing for explaining the join processing in the memory module group concerning the gestalt of the 4th operation.
[Drawing 45] Drawing 45 is drawing showing roughly connection of the memory module group in other applications of this invention.
[Description of Notations]
10 Computer System
12 CPU Module
14 Memory Module
16 Fixed Memory
18 Input Unit
20 Display
22 Legacy Memory
24 26 Bus
25 Control Signal Line
28, 29, 30 Switch
32 Clock Buffer
34 RAM Core
36 MPU
38 I/O

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12) **公 開 特 許 公 報**（Ａ）　(11)特許出願公開番号
特開2001－147800
（P2001－147800A）
(43)公開日　平成13年５月29日(2001.5.29)

(51)Int.Cl.^{7}	識別記号	ＦＩ	テーマコード゛（参考）
Ｇ０６Ｆ　7/24		Ｇ０６Ｆ　7/24	Ａ
12/00	５６０	12/00	５６０Ｆ
12/06	５３０	12/06	５３０Ａ
15/16	６２０	15/16	６２０Ａ
17/30		15/40	３１０Ｂ

審査請求　未請求　請求項の数14　ＯＬ　（全 34 頁）　最終頁に続く

(21)出願番号　特願平11－330997
(22)出願日　平成11年11月22日(1999.11.22)

(71)出願人　599074648
ターボデータラボラトリー有限会社
東京都台東区松が谷１－９－12　ＳＰＫビルディング604号
(72)発明者　古庄　晋二
神奈川県横浜市神奈川区松見町４丁目1101番地７コートハウス菊名804号
(74)代理人　100103632
弁理士　窪田　英一郎　（外１名）

(54)【発明の名称】　情報処理システム、並びに、この情報処理システムを利用したソート方法、コンパイル方法およびジョイン方法

(57)【要約】
【課題】　著しく高速に、かつ、安定した処理時間で、配列のソートなどを実現する。
【解決手段】　分散メモリ型の情報処理装置において、提示メモリモジュール14－1、14－3が自己のメモリモジュール内でソートされた要素を順位番号とともに、スイッチ３０などにより分割されたバス２４を介して、判定メモリモジュール14－2、14－4に与える。判定メモリモジュールは、与えられた順位番号に基づき、受理した要素の順位番号の候補を示す仮想順位番号を算出して、当該仮想順位番号を、他のバス２４を介して、提示メモリモジュールに返送する。提示メモリモジュールは、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位番号を更新する。

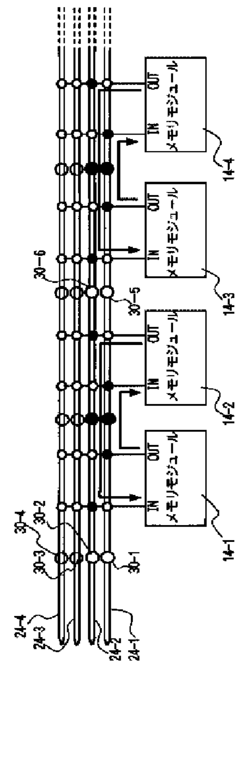


図8

【特許請求の範囲】
【請求項１】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムであって、
前記メモリモジュールのＭＰＵが、自己の把握する配列の部分を構成する要素のソートを実行して、前記要素を特定の順序にしたがって並べ替えるソート手段と、
前記自己の把握する前記部分が配列中に占める位置にしたがって、前記ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達し、或いは、所定のバスを介して、他のメモリモジュールからの前記要素および順位番号を受理するＩ／Ｏと、
前記要素および順位番号を受理した場合に、自己の把握する要素との比較により、受理した要素の順位番号の候補である仮想順位番号を算出して、前記他のメモリモジュールに返送する順位番号算出手段と、
前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位を確定する順位確定手段とを備え、
前記要素および順位番号を送出する側のメモリモジュールである提示メモリモジュールと、前記要素および順位番号受理して仮想順位番号を算出する側のメモリモジュールである判定メモリモジュールとの通信により、前記配列の要素の順位番号を確定することを特徴とする情報処理システム。
【請求項２】　前記メモリモジュールが、
確定した順位番号にしたがって、処理対象となる要素を特定して何れかのバスに送出する要素特定／送出手段と、
前回の処理対象となった要素と送出された要素とを比較する要素比較手段と、
同一の要素が送出された場合には、その値をカウントアップする、同一の要素の存在数を示す同一値個数カウンタとを備え、前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて何れかの送出するように構成され、さらに、
何れかのメモリモジュールが、
送出された前回の処理対象となった要素および関連するカウンタの値を受理して、これらを関連付け、かつ、受理した順序で配置する配列を備えたことを特徴とする請求項１に記載の情報処理システム。

【請求項３】　前記メモリモジュールが、
前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、その値をカウントアップする、重複のない順位番号を示す値番号カウンタと、
送出された要素に関して、前回の処理対象となった要素と送出された要素とが同一の場合には値番号カウンタの値を、重複のない当該要素の順位番号と決定し、その一方、これらが異なる場合には、カウントアップされた値番号カウンタの値を、重複のない当該要素の順位番号と決定して、当該順位番号を更新する順位番号更新手段とを備えたことを特徴とする請求項２に記載の情報処理装置。
【請求項４】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した配列のソート方法であって、（ａ）メモリモジュールにおいて、自己が把握する配列の部分を構成する要素をソートするステップと、（ｂ）前記自己の把握する前記部分が配列中に占める位置にしたがって、前記配列の部分を把握するメモリモジュールのうち、要素および順位番号を送出する側の提示メモリモジュール、および、要素および順位番号を受理する側の判定メモリモジュールを決定するステップと、（ｃ）提示メモリモジュールにおいて、ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達するステップと、（ｄ）判定メモリモジュールにおいて、所定のバスを介して他のメモリモジュールからの前記要素および順位番号を受理するステップと、（ｅ）前記判定メモリモジュールにおいて、当該判定メモリモジュールが把握する要素の順位番号に基づき、受理した要素の順位番号の候補を示す仮想順位番号を算出して、当該仮想順位番号を、前記提示メモリモジュールに返送するステップと、（ｆ）前記提示メモリモジュールにおいて、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位番号を更新するステップと、（ｇ）前記ステップ（ｄ）～（ｆ）が終了するごとに、当該ステップ（ｄ）～（ｆ）により所定の順位番号が付された要素に関する提示メモリモジュールと判定メモリモジュールからなるメモリモジュール群を、それぞれ、提示メモリモジュール群、および、判定メモリモジュール群の一方として、ステップ（ｄ）～（ｆ）を繰り返すことにより、各メモリモジュール群における要素の順位番号を更新することにより、配列の各要素の順位番

号を確定することを特徴とするソート方法。

【請求項５】　前記ステップ（ｅ）が、（e1）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号とに基づき、仮想順位番号を算出するステップを含むことを特徴とする請求項４に記載のソート方法。

【請求項６】　前記ステップ（ｆ）が、（f1）受理した仮想順位番号を、ステップ（ｃ）にて送出した要素の順位番号に代入するステップを含むことを特徴とする請求項４または５に記載のソート方法。

【請求項７】　さらに、（ｈ）提示メモリモジュール群において、当該提示メモリモジュール群を構成するメモリモジュールにて把握されている要素が、当該メモリモジュール群においていくつ存在しているかを示す重複数を算出するステップを備え、

前記ステップ（ｃ）が、（c1）同一の要素を重複して伝達しないように、ソートされた要素を、その順位番号および重複数とともに、他のメモリモジュールに伝達するステップを含み、前記ステップ（ｅ）が、（e2）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号および重複数とに基づき、仮想順位番号を算出するステップを含み、かつ、

前記ステップ（ｆ）が、（f2）仮想順位番号と、ステップ（ｃ）における要素の送出時の順位番号との差に基づき、当該要素と同一の要素の順位番号を決定するステップを含むことを特徴とする請求項４に記載のソート方法。

【請求項８】　初期的に提示メモリモジュールが単独のメモリモジュールであり、かつ、受信モジュールも単独のメモリモジュールであり、

（ｄ）～（ｆ）のステップが終了する毎に、ｎ（ｎ：１以上の整数）がインクリメントされるような2^nのメモリモジュールからなる提示メモリモジュール群と、2^nのメモリモジュールからなる判定メモリモジュール群が形成されることを特徴とする請求項４ないし７の何れか一項に記載のソート方法。

【請求項９】　請求項４ないし６の何れか一項に記載された方法により、配列をソートし、かつ、当該ソートされた配列に基づき、前記配列中の要素が、重複なく、かつ、所定の順序にて配置された新たな配列を生成するコンパイル方法であって、（ｉ）所定のメモリモジュールにおいて、順位番号にしたがって処理対象となる要素を送出するステップと、（ｊ）前回の処理対象となった要素と同一の要素が送出された場合には、同一の要素の存在数を示す同一値個数カウンタをカウントアップし、その一方、前回の処理対象となった要素と異なる要素が送出された場合には、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて、これらを送出するステップと、（ｋ）前回の処理対象となった要素、および、関連する同一値カウンタの値を受理して、これらを関連付けて新たな配列中に配置するステップとを備え、（ｌ）ステップ（ｉ）～（ｊ）を繰り返すことにより、前記新たな配列中に、要素およびその存在数が関連付けられて配置されることを特徴とするコンパイル方法。

【請求項１０】　さらに、（ｍ）何れかのモジュールにおいて、ステップ（ｊ）にて送出される要素および関連する同一値個数カウンタの値をモニターするステップを備え、

当該何れかのモジュールにより、ステップ（ｋ）が実行されることを特徴とする請求項９に記載のコンパイル方法。

【請求項１１】　（ｎ）当該配列の要素を把握するメモリモジュールにおいて、処理対象となっている要素の順位番号および当該要素の存在数をそれぞれ格納する順位番号カウンタおよび同一値個数カウンタを設けるとともに、および、前回の処理対象となった要素を一時的に格納するレジスタを設けるステップと、（ｏ）順位番号にしたがって、当該順位番号が付された要素を把握するメモリモジュールにおいて、当該要素を第１のバスに送出するステップと、（ｐ）配列の要素を把握するメモリモジュールにおいて、受理した要素とレジスタの内容とを比較して、これらが一致する場合には、存在数をカウントアップする一方、これらが一致しない場合には、レジスタの内容および存在数カウンタの値を、第２のバスに送出した後に、レジスタの内容および存在数カウンタの値を更新するステップと、（ｑ）何れかのメモリモジュールにおいて、前記レジスタの内容および存在数カウンタの値を、それぞれ要素および当該要素の存在数として、配列中に配置するステップとを備えたことを特徴とする請求項９に記載のコンパイル方法。

【請求項１２】　ステップ（ｎ）が、さらに、（n1）処理対象となっている要素に関して、重複のない順位番号を格納する値カウンタを設けるステップを含み、

前記ステップ（ｐ）が、（p1）受理した要素とレジスタの内容とを比較して、これらが一致する場合に、当該処理対象となる要素の順位番号に、値番号カウンタの値を付与する一方、これらが一致しない場合に、値番号カウンタをカウントアップし、処理対象となる要素の順位番号に、カウントアップされた値番号カウンタの値を付与するステップを含むことを特徴とする請求項１１に記載のコンパイル方法。

【請求項１３】　請求項４ないし請求項８の何れか一項に記載のソート方法、および、請求項９ないし１２の何れか一項に記載のコンパイル方法を用いて、複数の配列の共有化を実現する配列のジョイン方法であって、

（ｒ）複数の配列を合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行す

るステップと、（ｓ）前記合併した配列中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな配列を生成するステップとを備えたことを特徴とする配列のジョイン方法。

【請求項１４】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した複数の配列のジョイン方法であって、請求項４ないし請求項８の何れか一項に記載のソート方法、および、請求項９ないし１２の何れか一項に記載のコンパイル方法を用いて、複数の配列の共有化を実現する配列のジョイン方法において、

前記メモリモジュールが、それぞれ、レコード番号に基づき、要素を格納した配列である値リストにおける所定の要素を指定するために、レコード番号に対応する位置に、値リストを示すポインタ値を配置したポインタ配列を備え、（r1）複数の値リストを合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行するステップと、（ｔ）前記合併した値リスト中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな値リストを生成するとともに、前記要素の順位番号を、重複した要素の存在しない場合の当該要素の順位番号に更新するステップと、（ｕ）前記重複した要素の存在しない場合の要素の順位番号からなる配列を、新たな値リストを示すための、新たなポインタ配列とするステップとを備えたことを特徴とするジョイン方法。

【発明の詳細な説明】

【０００１】

【産業上の技術分野】本発明は、分散メモリ型の情報処理装置に関し、より詳細には、極めて高速にソート、コンパイルおよびジョインの処理を実現可能な情報処理装置に関する。

【０００２】

【従来の技術】社会全体のさまざまな場所にコンピュータが導入され、インターネットをはじめとするネットワークが浸透した今日では、そこここで、大規模なデータが蓄積されるようになった。このような大規模データを処理するには、膨大な計算が必要で、そのために並列処理を導入しようと試みるのは自然である。

【０００３】さて、並列処理アーキテクチャは「共有メモリ型」と「分散メモリ型」に大別される。前者（「共有メモリ型」）は、複数のプロセッサが１つの巨大なメモリ空間を共有する方式である。この方式では、プロセッサ群と共有メモリ間のトラフィックがボトルネックとなるので、百を越えるプロセッサを用いて現実的なシステムを構築することは容易ではない。したがって、例えば１０億個の浮動小数点変数の平方根を計算する際、単一ＣＰＵに対する加速比は、せいぜい１００倍ということになる。経験的には、３０倍程度が上限である。

【０００４】後者（「分散メモリ型」）は、各プロセッサがそれぞれローカルなメモリを持ち、これらを結合してシステムを構築する。この方式では、数百～数万ものプロセッサを組み込んだハードウェアシステムの設計が可能である。したがって、上記１０億個の浮動小数点変数の平方根を計算する際の単一ＣＰＵに対する加速比を、数百～数万倍とすることが可能である。

【０００５】

【発明が解決しようとする課題】数百を越える多数のプロセッサによる並列処理の潜在的需要は大きいといわれているが、上述したように、現在の現実的なハードウェア技術でこれを実現しようとすると、分散メモリ型以外の手法による設計は困難である。分散メモリ型においては、個々のプロセッサに付属するメモリの容量が小さいため、並列処理の主たる目的の一つである大規模データ（通常は、配列）の保持および処理において、これを複数プロセッサおよびそれぞれに付属するメモリが分掌する必要がある。

【０００６】しかしながら、複数プロセッサおよびそれぞれに付属するメモリが配列を分掌する場合には、バス上のデータの衝突を防止するためのバス調停が困難であり、各プロセッサが並列的に作動できなければ、プロセッサの利用効率を向上できず、その結果、処理の高速化を図ることができないなどの問題点がある。そこで、本発明は、以下のように、種々の目的を達成する。

【０００７】（１）アルゴリズム的にバス上のデータの衝突が発生せず、バス調停が不要であり、これにより、バスのバンド幅をフルに生かして処理速度を向上させる。

（２）プロセッサ（好ましくは複数のプロセッサ）と目盛りとを備えたメモリモジュールを多数組み合わせて、これらによる並列処理を可能とし、それぞれのメモリモジュールを有効利用し、かつ、各メモリモジュール内のプロセッサに独立した処理を割り当てられるようにし、これにより、メモリモジュールの有効利用により処理速度をさらに向上させる。

（３）ソート対象のデータの大きさを「Ｎ」とした場合に、Ｏ（Ｎ）のデータの大きさしか必要としない。（従来のソート処理では、最悪の場合にＯ（Ｎ＊Ｎ）やＯ（Ｎ＊Log(Ｎ)）のデータ量を必要とし得る。）

（４）処理時間が安定しており、最悪の場合でも、予想可能な処理速度が保証される。つまり、本発明は、著し

く高速に、かつ、安定した処理時間で、配列のソートなどが可能な情報処理装置を提供することを目的とする。
【０００８】
【課題を解決するための手段】本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムであって、前記メモリモジュールのＭＰＵが、自己の把握する配列の部分を構成する要素のソートを実行して、前記要素を特定の順序にしたがって並べ替えるソート手段と、前記自己の把握する前記部分が配列中に占める位置にしたがって、前記ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達し、或いは、所定のバスを介して、他のメモリモジュールからの前記要素および順位番号を受理するＩ／Ｏと、前記要素および順位番号を受理した場合に、自己の把握する要素との比較により、受理した要素の順位番号の候補である仮想順位番号を算出して、前記他のメモリモジュールに返送する順位番号算出手段と、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位を確定する順位確定手段とを備え、前記要素および順位番号を送出する側の提示メモリモジュールと、前記要素および順位番号受理して仮想順位番号を算出する側の判定メモリモジュールとの通信により、前記配列の要素の順位番号を確定することを特徴とする情報処理システムにより達成される。
【０００９】本発明によれば、提示メモリモジュールによる要素および順位番号の提示があるバスを介して実行され、判定メモリモジュールにより仮想順位番号が算出され、当該仮想順位番号が、他のバスを介して、提示メモリモジュールに与えられる。したがって、提示メモリモジュールおよび判定メモリモジュールにおいて、並列的にソート処理を進めることができ、かつ、バスの衝突も回避することが可能となる。
【００１０】本発明の好ましい実施態様においては、前記メモリモジュールが、確定した順位番号にしたがって、処理対象となる要素を特定して何れかのバスに送出する要素特定／送出手段と、前回の処理対象となった要素と送出された要素とを比較する要素比較手段と、同一の要素が送出された場合には、その値をカウントアップする、同一の要素の存在数を示す同一値個数カウンタとを備え、前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて何れかの送出するように構成され、さらに、何れかのメモリモジュールが、送出された前回の処理対象となった要素および関連するカウンタの値を受理して、これらを関連付け、かつ、受理した順序で配置する配列を備えている。この実施態様によれば、何れかのメモリモジュールにおいて、要素およびその重複数が、所定の順序で受理され、これにより、重複のない要素の配列、および、各要素の存在数を示す配列を作成することが可能となる。すなわち、これにより、重複のない要素のリスト、および、もとの配列において各要素の存在する数を容易に把握することができる。
【００１１】本発明の別の実施態様によれば、前記メモリモジュールが、前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、その値をカウントアップする、重複のない順位番号を示す値番号カウンタと、送出された要素に関して、前回の処理対象となった要素と送出された要素とが同一の場合には値番号カウンタの値を、重複のない当該要素の順位番号と決定し、その一方、これらが異なる場合には、カウントアップされた値番号カウンタの値を、重複のない当該要素の順位番号と決定して、当該順位番号を更新する順位番号更新手段とを備えている。この実施態様によれば、配列の要素に付与された順位番号を、要素の重複を排除した状態のものに変換することが可能となる。
【００１２】また、本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した配列のソート方法であって、（ａ）メモリモジュールにおいて、自己が把握する配列の部分を構成する要素をソートするステップと、（ｂ）前記自己の把握する前記部分が配列中に占める位置にしたがって、前記配列の部分を把握するメモリモジュールのうち、要素および順位番号を送出する側の提示メモリモジュール、および、要素および順位番号を受理する側の判定メモリモジュールを決定するステップと、（ｃ）提示メモリモジュールにおいて、ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達するステップと、（ｄ）判定メモリモジュールにおいて、所定のバスを介して他のメモリモジュールからの前記要素および順位番号を受理するステップと、（ｅ）前記判定メモリモジュールにおいて、当該判定メモリモジュールが把握する要素の順位番号に基づ

き、受理した要素の順位番号の候補を示す仮想順位番号を算出して、当該仮想順位番号を、前記提示メモリモジュールに返送するステップと、（ｆ）前記提示メモリモジュールにおいて、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位番号を更新するステップと、（ｇ）前記ステップ（ｄ）～（ｆ）が終了するごとに、当該ステップ（ｄ）～（ｆ）により所定の順位番号が付された要素に関する提示メモリモジュールと判定メモリモジュールからなるメモリモジュール群を、それぞれ、提示メモリモジュール群、および、判定メモリモジュール群の一方として、ステップ（ｄ）～（ｆ）を繰り返すことにより、各メモリモジュール群における要素の順位番号を更新することにより、配列の各要素の順位番号を確定することを特徴とするソート方法によっても達成される。

【００１３】上記発明によれば、提示メモリモジュールにおける演算、提示メモリモジュールにおける要素および順位番号の送出、判定メモリモジュール群における演算、提示メモリモジュールにおける仮想順位番号の送出が、並列的に実行でき、かつ、バスの衝突も回避することができる。すなわち、これにより、著しく高速に、ソート処理（配列の要素への順位番号付与）を実現することが可能となる。また、使用するメモリ量も、Ｏ（Ｎ）に抑制することが可能となる。上記発明の好ましい実施態様においては、ステップ（ｅ）が、（e1）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号とに基づき、仮想順位番号を算出するステップを含む。また、さらに好ましい実施態様においては、ステップ（ｆ）が、（f1）受理した仮想順位番号を、ステップ（ｃ）にて送出した要素の順位番号に代入するステップを含む。

【００１４】本発明の好ましい実施態様においては、さらに、（ｈ）提示メモリモジュール群において、当該提示メモリモジュール群を構成するメモリモジュールにて把握されている要素が、当該メモリモジュール群においていくつ存在しているかを示す重複数を算出するステップを備え、前記ステップ（ｃ）が、（c1）同一の要素を重複して伝達しないように、ソートされた要素を、その順位番号および重複数とともに、他のメモリモジュールに伝達するステップを含み、前記ステップ（ｅ）が、（e2）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号および重複数とに基づき、仮想順位番号を算出するステップを含み、かつ、前記ステップ（ｆ）が、（f2）仮想順位番号と、ステップ（ｃ）における要素の送出時の順位番号との差に基づき、当該要素と同一の要素の順位番号を決定するステップを含む。

【００１５】この実施態様によれば、提示メモリモジュールは、同じ要素を何度も送出する必要がない。また、

ある要素の重複数を算出すると、当該要素の順位番号および重複数を、判定メモリモジュールに送信し、判定メモリモジュールでは、当該要素にかかる仮想順位番号の算出を実行することができる。すなわち、これにより、メモリモジュールの利用効率を下げることを防止できる。

【００１６】さらに好ましい実施態様においては、初期的に提示メモリモジュールが単独のメモリモジュールであり、かつ、受信モジュールも単独のメモリモジュールであり、（ｄ）～（ｆ）のステップが終了する毎に、ｎ（ｎ：１以上の整数）がインクリメントされるような２^{n}のメモリモジュールからなる提示メモリモジュール群と、２^{n}のメモリモジュールからなる判定メモリモジュール群が形成される。上述したように、２^{n}のメモリモジュールを用いると、好適に、ソート処理を実現することが可能となる。

【００１７】また、本発明の別の実施態様においては、上記ソート方法により、配列をソートし、かつ、当該ソートされた配列に基づき、前記配列中の要素が、重複なく、かつ、所定の順序にて配置された新たな配列を生成するコンパイル方法は、（ｉ）所定のメモリモジュールにおいて、順位番号にしたがって処理対象となる要素を送出するステップと、（ｊ）前回の処理対象となった要素と同一の要素が送出された場合には、同一の要素の存在数を示す同一値個数カウンタをカウントアップし、その一方、前回の処理対象となった要素と異なる要素が送出された場合には、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて、これらを送出するステップと、（ｋ）前回の処理対象となった要素、および、関連する同一値カウンタの値を受理して、これらを関連付けて新たな配列中に配置するステップとを備え、（ｌ）ステップ（ｉ）～（ｊ）を繰り返すことにより、前記新たな配列中に、要素およびその存在数が関連付けられて配置されることを特徴とする。

【００１８】また、上記コンパイル方法は、さらに、（ｍ）何れかのモジュールにおいて、ステップ（ｊ）にて送出される要素および関連する同一値個数カウンタの値をモニターするステップを備え、当該何れかのモジュールにより、ステップ（ｋ）が実行されても良い。

【００１９】また、上記コンパイル方法は、（ｎ）当該配列の要素を把握するメモリモジュールにおいて、処理対象となっている要素の順位番号および当該要素の存在数をそれぞれ格納する順位番号カウンタおよび同一値個数カウンタを設けるとともに、および、前回の処理対象となった要素を一時的に格納するレジスタを設けるステップと、（ｏ）順位番号にしたがって、当該順位番号が付された要素を把握するメモリモジュールにおいて、当該要素を第１のバスに送出するステップと、（ｐ）配列の要素を把握するメモリモジュールにおいて、受理した

要素とレジスタの内容とを比較して、これらが一致する場合には、存在数をカウントアップする一方、これらが一致しない場合には、レジスタの内容および存在数カウンタの値を、第２のバスに送出した後に、レジスタの内容および存在数カウンタの値を更新するステップと、
（ｑ）何れかのメモリモジュールにおいて、前記レジスタの内容および存在数カウンタの値を、それぞれ要素および当該要素の存在数として、配列中に配置するステップとを備えても良い。
【００２０】ステップ（ｎ）は、さらに、（n1）処理対象となっている要素に関して、重複のない順位番号を格納する値カウンタを設けるステップを含み、前記ステップ（ｐ）は、（p1）受理した要素とレジスタの内容とを比較して、これらが一致する場合に、当該処理対象となる要素の順位番号に、値番号カウンタの値を付与する一方、これらが一致しない場合に、値番号カウンタをカウントアップし、処理対象となる要素の順位番号に、カウントアップされた値番号カウンタの値を付与するステップを含むのが、さらに望ましい。
【００２１】また、本発明の別の実施態様において、上記ソート方法および上記コンパイル方法を用いて、複数の配列の共有化を実現する配列のジョイン方法は、
（ｒ）複数の配列を合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行するステップと、（ｓ）前記合併した配列中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな配列を生成するステップとを備えている。すなわち、所望の配列を併合した状態で、本発明にかかるソート方法およびコンパイル方法を施すことにより、要素の重複を排除した、ジョインされた配列を得ることが可能となる。
【００２２】さらに別の実施態様において、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した複数の配列のジョイン方法は、
【００２３】上記ソート方法および上記コンパイル方法を用いており、かつ、メモリモジュールが、それぞれ、レコード番号に基づき、要素を格納した配列である値リストにおける所定の要素を指定するために、レコード番号に対応する位置に、値リストを示すポインタ値を配置したポインタ配列を備え、（r1）複数の値リストを合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行するステップと、（ｔ）前記合併した値リスト中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな値リストを生成するとともに、前記要素の順位番号を、重複した要素の存在しない場合の当該要素の順位番号に更新するステップと、
（ｕ）前記重複した要素の存在しない場合の要素の順位番号からなる配列を、新たな値リストを示すための、新たなポインタ配列とするステップとを備えている。
【００２４】
【発明の実施の形態】［ハードウェア構成］以下、添付図面を参照して、本発明の実施の形態につき説明を加える。図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。図１に示すように、コンピュータシステム１０は、単一命令による並列演算を実現するＣＰＵモジュール１２と、並列演算のために必要な種々のデータを記憶するメモリモジュール１４－１、１４－２、１４－３、…と、必要なプログラムやデータを記憶する固定記憶装置１６と、キーボードやマウスなどの入力装置１８と、ＣＲＴなどからなる表示装置２０と、種々の形式のデータ等が記憶されているレガシーメモリ２２とを備えている。また、バス２４－１、２４－２、…において、ＣＰＵモジュール１２、各メモリモジュール１４との接点には、スイッチ２８－１、２８－２、２８－３、…などが配設され、選択された回路要素間における情報の授受が可能となっている。また、ＣＰＵモジュール１２とメモリモジュール１４－１との間、隣接するメモリモジュール間において、バスの連結および接続をなすためのスイッチ３０－１、３０－２、…が設けられている。また、メモリモジュールの入力端子とバスとの接点と、当該メモリモジュールの出力端子とバスとの接点との間に、スイッチ（符号２９参照）が設けられていても良い。図１において、上記スイッチは、破線の丸印にて示されている。
【００２５】さらに、メモリモジュール１４には、単一の入力端子および単一の出力端子だけではなく、一以上の他の端子（入出力端子など）が設けられているのが望ましい。たとえば、後述する第２の実施の形態や第３の実施の形態においては、３つ以上の端子からの入出力を用いて処理が実現されている。
【００２６】ＣＰＵモジュール１２と、メモリモジュール１４との間には、複数のバス２４－１、２４－２、２４－３、２４－４、…とが設けられている。したがって、ＣＰＵモジュール１２とメモリモジュール１４との間、および、メモリモジュール間は、上記バスによりデータ等の授受が可能となっている。また、ＣＰＵ１２と、メモリモジュール１４との間には、制御信号ライン２５が設けられ、ＣＰＵ１２から発せられるインストラクションなどが、全てのメモリモジュール１４に伝達されるようになっている。
【００２７】さらに、ＣＰＵ１２と、他の構成要素（た

とえば、固定記憶装置１６、入力装置１８など）との間には、ローカルバス２６が配設されており、これらの間でもデータ等の授受が可能となっている。ＣＰＵ１２は、固定記憶装置１６に記憶され、或いは、バス２６上に接続されたＲＡＭのような他の記憶装置（図示せず）に記憶されたプログラムを読み出し、このプログラムにしたがって、以下に示すメモリモジュール１４へのインストラクションの送出を含むデータの授受のほか、スイッチ２８～３０の制御等を実行する。また、ＣＰＵ１２は、プログラムにしたがって、レガシーメモリ２２に記憶された種々の形式のデータを受け入れて、この形式のデータを、ＣＰＵ１２、メモリモジュール１４、バス２４からなる系にて処理可能な一連のデータ（配列）に変換し、これらを、各メモリモジュール１４に記憶させることもできる。

【００２８】図２は、各メモリモジュール１４の概略を示すブロックダイヤグラムである。図２に示すように、メモリモジュール１４は、ＣＰＵモジュール１２から与えられるクロックなど同期信号を受け入れるクロックバッファ３２と、データを記憶するＲＡＭコア３４と、後述する空間ＩＤやデータの要素番号等を把握し、ＣＰＵ１２からのインストラクションなどを受理した場合に、空間ＩＤや要素番号に基づき、ＲＡＭコア３４へのデータ書き込みやＲＡＭコアからのデータ読み出しを制御するＭＰＵ３６と、バスの何れかからのデータを受け入れて、ＲＡＭコア３４に供給し、および／または、ＲＡＭコア３４からのデータを何れかのバスに送出するＩ／Ｏ３８とを有している。この実施の形態において、メモリモジュール１４は、制御信号ライン２５を介して、ＣＰＵからのインストラクションを受け入れ、ＭＰＵ３６が、このインストラクションに応答して、ＲＡＭコア３４のデータを読み出し、ＲＡＭコア３４にデータを書き込み、或いは、データに所定の処理を施すことができるようになっている。また、ＲＡＭコア３４へのデータアクセスや、Ｉ／Ｏを介してデータ入力およびデータ出力は、クロックバッファ３２に与えられるクロックなどの同期信号に基づき実行される。上記メモリモジュール１４のＭＰＵ３６は、複数の処理ユニットからなり、並列的に複数の処理を実行できるのが望ましい。

【００２９】図１および図２から明らかなように、本発明において、コンピュータシステム１０は、メモリ共有型のシステムであると考えることができる。また、後述するように、制御信号ライン２５を介して、各メモリモジュール１４にインストラクションを与えることにより、各メモリモジュール１４が並列的に処理を実行する。また、バスへのデータ出力およびバスからのデータ入力などが、所定の同期信号に基づき実行される。したがって、このコンピュータシステム１０は、ＳＩＭＤの形態をなしていると考えることができる。このような構成を備えたコンピュータシステム１０は、基本的には、本発明者の考案にかかる、特願平１１－２６３７９３号に記載された多空間メモリ、メモリモジュールおよび組み替え可能バスを備えている。これらにつき、以下に簡単に説明を加える。

【００３０】（１）多空間メモリ
本明細書において、多空間メモリとは、メモリ空間を、空間ＩＤとアドレスとに基づきアクセスするために割り当てられたメモリ空間をいう。これにより、一連のデータが多数のプロセッサに分掌されていても、各プロセッサが、これを確実に分離、認識することができる。

【００３１】従来のメモリ空間においては、プロセス毎に個別の領域を割り当てることはあっても、一連の変数（配列、構造体など）毎にメモリ空間を割り当てることは行われてこなかった。したがって、以下、このような従来のメモリ空間を「単一メモリ空間」と称する。単一メモリ空間のシステムにおいては、アドレスのみを用いてデータにアクセスしているため、関連を有する一連のデータを分離したり、認識することができなかった。このため、実際には並列処理が可能であっても、その可否を判断できない場合が多かった。また、ある単一メモリ空間に、新たな一連のデータを収容させる場合に、当該一連のデータの収容場所を確保するために、ガーベージコレクションを実行する必要があった。

【００３２】これに対して、本発明においては、メモリ空間に、空間ＩＤを導入し、一連のデータについて同一のＩＤを付与している。また、メモリモジュール１４において、自身のＲＡＭコア３４に保持されているデータに関する空間ＩＤを把握し、これにより、各メモリモジュール１４自体が、現在アクセスされているデータの空間ＩＤを参照することにより、自己の作動の是非を決定することができる。また、各メモリモジュールが空間ＩＤと関連付けて、一連のデータの全部或いは一部を保持できるため、ある一連のデータを、複数のメモリモジュール１４に分割して記憶させることができ、これによりガーベージコレクションを不要にすることができる。

【００３３】（２）メモリモジュール
また、本発明においては、各メモリモジュール１４が、ＭＰＵ３６を有し、上記空間ＩＤのほか、自己が保持する一連のデータの各々の要素番号を把握している。したがって、ＣＰＵ１２からのインストラクションを受理した後、ＭＰＵ３６が、インストラクションにしたがってアクセスすべきデータが、自己のＲＡＭコア３４中に保持されているものか否かを判断して、アクセスに必要の是非を決定することができる。さらに、各メモリモジュール１４が、自己のＲＡＭコア３４に格納されている配列要素の添え字の範囲から、ＳＩＭＤでのインストラクションにおける暗黙の処理の分担範囲を決定することが可能である。各メモリモジュール１４は、ＣＰＵ１２からのインストラクションにしたがって、処理すべき要素の記憶順序を入れ替えて、自己のＲＡＭコア３４中に保

持している要素をソートすることが可能である。

【００３４】（３）組み替え可能バス
本発明においては、ＣＰＵ１２が、スイッチ２８－１、２８－２、…およびスイッチ３０－１、３０－２、…を選択的にオン／オフして、データの授受をなすべきメモリモジュール１４を指定することにより、パイプライン処理を実現している。たとえば、図３に示すように、あるメモリモジュール１４－ｉから出力されたデータを、他のメモリモジュール１４－ｊに与え、かつ、当該他のメモリモジュール１４－ｊから出力されたデータを、さらに他のメモリモジュール１４－ｋに伝達すべき場合には、ＣＰＵ１２は、バス２４－ｍを、メモリモジュール１４－ｉ、１４－ｊのために割り当て、かつ、バス２４－ｎを、メモリモジュール１４－ｊ、１４－ｋのために割り当てるように、各スイッチの状態を設定する。

【００３５】さらに、これらパイプライン処理は、単一のメモリモジュール間の接続により実現される場合だけでなく、複数の一連のメモリモジュール（メモリモジュール群）の間の接続により実現することも可能である。達成しようとする処理に応じて、各メモリモジュール間をつなぎ替え、各接続経路毎に、定められた種類のデータを定められた順序にて一方向に連続転送することで、バスの能力を１００％近く使用できるように、通信をスケジュール化することができる。これにより、分散メモリ型の並列処理システムの最大の問題であった、プロセッサ間通信のパフォーマンスの低さを、解消することができる。

【００３６】［多空間メモリ］再度、多空間メモリを用いた、本発明にかかるコンピュータシステムにおける各メモリモジュールのメモリ管理、および、インストラクションにしたがったメモリアクセスにつき、より詳細に説明を加える。図４は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。図４（ａ）に示すように、メモリモジュール１４中のＲＡＭコア３４には、空間ＩＤ管理テーブルが設けられる。これにより、メモリモジュール１４のＭＰＵ３６は、自己が保持するデータの空間ＩＤ等必要な情報を把握することが可能となる。

【００３７】図４（ｂ）に示すように、空間ＩＤ管理テーブルには、自己が保持するデータ群ごとの、空間ＩＤ、ＣＰＵの管理の下での、データ群の論理開始アドレス、データ群が割り付けられた領域のサイズ、ＲＡＭコア３４中の物理開始アドレス、当該空間ＩＤを有する一連のデータの全サイズ、および、アクセス制限を示すアクセス制限フラグが格納されている。アクセス制限フラグは、この実施の形態においては、読み出しのみ可能（Ｒ）、書き込みのみ可能（Ｒ）、読み書き可能（ＲＷ）の３つの状態を示すことができるようになっている。メモリモジュール１４のＭＰＵ３６は、ある空間ＩＤを有するデータ群が与えられた際に、ＲＡＭコア３４

中に当該データ群を収容すべき、１以上の領域を見出して、当該領域にデータ群をそのまま、或いは、２以上に分割して収容する。この際に、与えられた空間ＩＤ、論理開始アドレス、全サイズ、アクセス制限フラグとともに、実際にデータを収容したＲＡＭコア中の論理開始アドレスや、割り付け領域サイズも、空間ＩＤ管理テーブルに記憶される。図４（ｃ）は、図４（ｂ）による空間ＩＤ管理テーブルにしたがったＲＡＭコア３６中のデータを示す図である。

【００３８】［メモリアクセスの概略］このように構成されたメモリモジュール１４へのアクセスにつき以下に説明を加える。図５に示すように、まず、ＣＰＵ１２が、空間ＩＤおよび論理アドレス、並びに、必要なインストラクション（たとえば、データの書き込みや読み出し）を、制御信号ライン２５を介して、全てのメモリモジュール１４に伝達する。各メモリモジュール１４においては、これに応答して、ＭＰＵ３６に設けられた空間コンパレータ５２が、空間ＩＤと、自己の空間ＩＤ管理テーブル上に保持されている空間ＩＤとを比較して、同一のものを、自己が保持しているかを判断し、また、、アドレスコンパレータ５４が、論理アドレスについて、同様の判断を行う。次いで、メモリモジュール１４のＭＰＵ３６が、自己のＲＡＭコア３４に、インストラクションによる処理対象となるデータが保持されていると判断した場合には、アドレスカリキュレータ５６が、空間ＩＤ管理テーブルを参照して、ＲＡＭコア３４中の物理アドレスを算出し、処理対象となるデータを特定する。このようにして、データが特定された後に、ＭＰＵ３６は、ＣＰＵ１２から与えられたインストラクションに応じた処理（たとえば、データの書き込みや読み出し）を実行し、必要な場合には、データをＣＰＵ１２に伝達する（図５（ｃ）参照）。

【００３９】［ソート処理（第１の実施の形態）］このように構成されたコンピュータシステム１０にかかるソート処理につき説明を加える。なお、以下の説明において、本発明にかかるメモリモジュールが、ＭＰＵ（プロセッサ）を備えたメモリモジュールであることから、ＰＭＭ（Processor Memory Module）と称する。

【００４０】理解を容易にするために、図６に示すように、４つのＰＭＭが、それぞれ、２つの要素（名字）を保持している場合を考える。図６（ａ）に示すように、あるＰＭＭ（第１のＰＭＭ１４－１）には、要素の添え字（すなわちレコード番号）が「０」である「やまもと」という姓と、添え字が「１」である「あべ」という姓とが保持されている。第２のＰＭＭ１４－２には、添え字が「２」である「いとう」という姓と、添え字が「３」である「すぎもと」という姓とが保持されている。以下、第３のＰＭＭ１４－３、第４のＰＭＭ１４－４にも、それぞれ、図６（ａ）に示すような添え字に対応した姓が保持されている。これら要素からなる配列に

は、同一の空間ＩＤが付され、各ＰＭＭのＭＰＵ３６は、その空間ＩＤ管理テーブルを利用して、自己のＲＡＭコア３４が管理する要素の添え字（レコード番号）や実際に格納している物理アドレス等を管理している。
【００４１】たとえば、ＣＰＵ１２から、制御信号ライン２５を介して、この空間ＩＤを有する配列のソートをするインストラクションが、各ＰＭＭ１４－１～１４－４に与えられたと考える。図７は、本実施の形態にかかるソート処理の処理手順を示すフローチャートである。図７に示すように、インストラクション（たとえば、「ある空間ＩＤを有する配列中の要素をソートせよ」というインストラクション）がＣＰＵ１２により発行されると（ステップ７００）、このインストラクションに応答して、各ＰＭＭにおいて、各ＰＭＭのＭＰＵ３６は、制御信号ライン２５を介して与えられたインストラクションを受理して、その内容を解釈し（ステップ７０１）、インストラクション中の「空間ＩＤ」を調べ（ステップ７０２）、自己のＲＡＭコア３４が保持するデータの空間ＩＤに関連しているか否かを判断する（ステップ７０３）。ステップ７０３にてノー（No）と判断された場合には、処理を終了し、その一方、イエス（Yes）と判断された場合には、ＭＰＵ３６は、空間ＩＤ管理テーブルを参照して、当該空間ＩＤに関するデータ群が書き込み可能な状態になっているかなど、必要なチェックを行う（ステップ７０４）。チェックによって異常があると判断された場合（ステップ７０５でイエス(Yes)）には、ＭＰＵ３６は、制御信号ライン２５を介してエラーが生じたことをＣＰＵ１２に通知する。その一方、異常がない場合には、ＭＰＵ３６は、以下に述べるソート処理本体を実行する（ステップ７０７以下）。
【００４２】まず、処理に関連するＰＭＭ１４－１～１４－４の各々は、自己の保持する要素のソートを実行する（ステップ７０７）。このソートは、実際に、各ＰＭＭ１４中の要素の入れ替えを伴う。より具体的には、ＭＰＵ３６が、自己のＲＡＭコア３４中に保持された要素を、クイックソートなど既知のソート手法を用いてソートする。図６（ｂ）は、図６（ａ）に示す各ＰＭＭ中の配列中の要素がソートされた状態を示す図である。なお、図６（ｂ）に示すように、上記要素のソートにともなって、各要素の添え字（レコード番号）の配置も変更されていることに留意すべきである。
【００４３】次いで、各ＰＭＭ１４のＭＰＵ３６は、自己が保持／管理している配列中の要素の数だけ、順位番号を配置するための領域（順位番号領域）を確保し、各順位番号の初期値を与える（ステップ７０８）。図６（ｃ）は、各ＰＭＭに関して、順位番号の初期値が与えられた状態を示す図である。このように、初期的に、順位番号は、各モジュール内にてソートされた要素内で付与される。次いで、隣接するペア間のマージおよび順位番号付与が実行される（ステップ７０９）。ステップ７０９において、まずＣＰＵ１２が、バス２４上のスイッチ２８、３０を制御して、ソート処理に関連するＰＭＭのうち、所定のペアの一方の入力と他方の出力、並びに、一方の出力と他方の入力とを接続する。上記ペアは隣接している２つのＰＭＭ、隣接していない場合にも近傍に位置する２つのＰＭＭからなるのが望ましい。たとえば、図１において、ソート処理に関連するものが、ＰＭＭ１４－１～１４－４である場合には、ＰＭＭ１４－１および１４－２をペアとして、かつ、ＰＭＭ１４－３および１４－４をペアとするのが望ましい。ＣＰＵ１２は、たとえば、図８に示すように、バス２４－１にＰＭＭ１４－１の出力およびＰＭＭ１４－２を接続するように、かつ、バス２４－２にＰＭＭ１４－１の入力とＰＭＭ１４－２の出力とを接続するように、スイッチ２８を制御し、かつ、バス２４－１にＰＭＭ１４－３の出力およびＰＭＭ１４－４を接続するように、かつ、バス２４－２にＰＭＭ１４－３の入力とＰＭＭ１４－４の出力とを接続するように、スイッチ２８を制御する。さらに、ＣＰＵ１２は、さらに、ＰＭＭ１４－２とＰＭＭ１４－３との間に配置されるバス２４－１、２４－２上のスイッチ３０－５、３０－６をオフにする。図８において、黒丸で表わしているものが導通している状態を示し、白丸で表わしているものが導通ないしＰＭＭと接続されていない状態を示している。また、他のものは他のＰＭＭ（図示せず）の状態に従っている。なお、図８の例では、バス２４－１、２４－２を、スイッチ３０－５、３０－６をオフにすることにより分割して、バスをより有効に利用していることが理解できるであろう。
【００４４】このようにして、図９に模式的に示すように、ＣＰＵ１２によってＰＭＭ間の接続が規定されると、ＰＭＭのペア間での順位番号付与の処理本体が実行される。図１０ないし図１２は、理解を容易にするために図６にて示した配列に関する順位番号付与を模式的に示す図であり、図１３は、より一般的な、ＰＭＭのペア間の順位番号処理を示すフローチャートである。　図１０ないし図１２においては、ＰＭＭ１４－１およびＰＭＭ１４－２における処理過程のみを示したが、ＰＭＭ１４－３およびＰＭＭ１４－４における処理も、並列的に実行されている。なお、ここで、処理において、最初にデータを他方のＰＭＭに与えるものを前半のＰＭＭと称し、受理するもの（他方のＰＭＭ）を後半のＰＭＭと称する。前半のＰＭＭは、要素や順位番号を提示するため提示ＰＭＭということができ、その一方、後半のＰＭＭは、提示された順位番号を判定するため判定ＰＭＭということができる。ペアのうち何れのＰＭＭが前半のＰＭＭとなっても良い。この例では、便宜的に、ＰＭＭ１４－１が前半のＰＭＭとなり、ＰＭＭ１４－２が後半のＰＭＭとなっている。
【００４５】まず、前半のＰＭＭにおいては、処理位置を示すポインタ（以下、「ＰＵＴポインタ」と称する）

を初期位置（ソートされた配列の部分において先頭つまり「０」番目の位置）に配置する。その一方、後半のＰＭＭにおいて、以下に説明するように、前半のＰＭＭから受理する要素と、まず比較すべき位置などを示すポインタ（以下、「比較ポインタ」と称する）を初期位置（ソートされた配列の部分において先頭つまり「０」番目の位置）に配置する（図１０（ａ）、および、図１３のステップ１３０１、１３１１参照）。本実施の形態において、後半のＰＭＭにて使用する比較ポインタは、（Ｘ、Ｙ、Ｚ）という構造体配列の形態をとっている。ここに、Ｘは、比較すべき先頭位置（つまり、未比較の要素の先頭位置、以後「未処理位置」と称する）を示し、Ｙは、前半のＰＭＭから受信した要素の総数を示し（以下、場合によって「前方挿入数」と称する。）、Ｚは、前半のＰＭＭと後半のＰＭＭとをマージして得られた仮想的な配列における、前半のＰＭＭから与えられた要素の順位番号の案（以下、場合によって「仮想順位番号」と称する。）を示す。

【００４６】次いで、前半のＰＭＭのＭＰＵにより、最初のデータ転送が実行される。このデータ転送では、ＰＵＴポインタが示す位置の要素が、バスを介して、後半のＰＭＭに伝達される（図１０（ｂ）およびステップ１３０３、１３１２参照）。なお、ステップ１３０２の分岐では、２つのＰＭＭ間の処理では常にイエス（Yes）と判断されるが、これについては後述する。最初のデータ転送では、要素「あべ」が後半のＰＭＭに伝達される。後半のＰＭＭにおいては、後半のＰＭＭに格納されている配列の部分において、伝達された要素「あべ」を挿入すべき位置を捜し出す（ステップ１３１３）。これは実際に値を挿入するのではなく、挿入すべき位置を捜し出せば良い。本実施の形態において、各ＰＭＭの配列の部分に格納された要素は、実際にソートされた状態で配置されている。したがって、挿入位置の検索は、バイセクション法（二分割法）など、高速検索手法を用いて実現することができる。挿入位置を捜し出すことにより、順位が確定していない要素であって、挿入位置より前方に位置する要素の範囲（以下、「範囲１」と称する）を特定することが可能となる。なお、本実施の形態において、同じ要素があった場合には、前半のＰＭＭの順位が優先するという取り決めをしている。したがって、前半のＰＭＭから伝達された要素「あべ」が、後半のＰＭＭにも存在する場合には、前半ＰＭＭに格納されている方の順位が優先（つまり、より小さな順位番号）となる。

【００４７】本例では、前半のＰＭＭから伝達された要素「あべ」は、後半のＰＭＭが把握する配列の部分中、要素「あべ」の前方に位置することがわかり、これにより、範囲「１」に属する要素が存在しないことがわかる（図１０（ｃ）参照、および、ステップ１３１４においてイエス(Yes)）。そこで、後半のＰＭＭのＭＰＵは、伝達された要素「あべ」の順位番号として「０（すなわち先頭）」を、他方のバスを介して、前半のＰＭＭに返送する（ステップ１３１５）。次いで、後半のＰＭＭのＭＰＵは、前方挿入数をインクリメントして「１」にするとともに、仮想順位番号をインクリメントして「１」にする（ステップ１３１６）。これは、前方のＰＭＭから伝達された要素が一つ増加したため、前方挿入数をインクリメントする必要があり、かつ、次の要素の順位番号は、少なくとも、今回与えたもの（この場合では「０」）をインクリメントする必要があるからである。後半のＰＭＭから要素の順位番号（挿入位置）が与えられると（ステップ１３３２）、前半のＰＭＭのＭＰＵは、与えられた順位番号を、該当する要素の順位番号として格納し（ステップ１３３４）、次いで、ＰＵＴポインタをインクリメントする（図１１（ａ）、および、ステップ１３３５参照）。このようにして、前半のＰＭＭ中のある要素の順位が確定する。

【００４８】次に、前半のＰＭＭのＭＰＵは、ＰＵＴポインタが示す位置の要素「やまもと」を、バスを介して、後半のＰＭＭに伝達する（図１１（ｂ）およびステップ１３０３参照）。後半のＰＭＭにおいては、先に、要素「あべ」が伝達されたときと同様に、伝達された要素「やまもと」を挿入すべき位置を捜し出す（ステップ１３１３）。要素「やまもと」は、後半のＰＭＭが把握する配列の部分中、要素「はら」の後方に位置することがわかる（図１１（ｃ）、および、ステップ１３１４にてイエス(Yes)）。これにより、後半のＰＭＭが把握する配列の部分において、要素「はら」およびその前方に位置する要素の数、並びに、各要素の順位を確定することができる。より詳細には、後半のＰＭＭのＭＰＵは以下の手順にて、上記要素の順位を確定させる。

【００４９】まず、範囲「１」に含まれる要素に関する順位番号に、それぞれ、前方挿入数「Ｙ」が加えられる（ステップ１３１７）。これにより、範囲「１」に含まれる要素の順位が確定する。前述した例では、要素「あべ」の順位番号は「０＋１＝１」、要素「はら」の順位番号は「１＋１＝２」となる。次いで、範囲１に含まれる要素のうち、最後尾の要素の順位番号が、仮想番号に代入されるとともに（ステップ１３１８）、未処理位置を、範囲１における最後尾の要素の次の要素の位置に変更する（ステップ１３１９）。上記例では、要素「はら」の順位番号「２」が、比較ポインタ（構造体配列）のＺに与えられ、かつ、未処理位置が、「０」から「２」に変更される。これにより、構造体配列は（２、１、２）となる。このような処理の後、構造体配列中の、前方挿入数「Ｙ」および仮想順位番号「Ｚ」がインクリメントされる（ステップ１３２０）。これにより、構造体配列は、（２、２、３）となる（図１１（ｄ）参照）。ステップ１３２０にて得られた仮想順位番号が、ステップ１３１２にて受信した要素（上記例では、「や

まもと」）の順位番号となり、後半のＰＭＭのＭＰＵは、当該順位番号（上記例では「３」）を前半のＰＭＭに伝達する（ステップ１３２１）。このような処理の後に、さらに、仮想順位番号がインクリメントされる（ステップ１３２２）。これは、次の要素の順位番号は、少なくとも、今回与えた順位番号よりも１つ大きなものとなるからである。
【００５０】前半のＰＭＭは、受理した順位番号を、該当する要素の順位番号として格納し、次いで、ＰＵＴポインタをインクリメントする。このようにして、前半のＰＭＭにおける要素の順位番号が確定する。前半のＰＭＭにおいて、未処理の要素が既に存在しない（すなわち、すべての要素について順位番号が確定し、ＰＵＴポインタの位置には要素が配置されていない）には、前半のＰＭＭのＭＰＵは、終了を示す値を、後半のＰＭＭに伝達する（ステップ１３０６参照）。ここに、終了を示す値は、配列の最後尾の要素を示す値よりも大きな値である。後半のＰＭＭは、上記終了を示す値の受理に応答して、略同様の処理（図１３のステップ１３１２～１３２２）の処理を実行する。上記例では、終了を示す値の受理にもかかわらず、範囲「１」に含まれる要素が存在しないため、ステップ１３１５、１３１６を介してステップ１３２３に達し、処理を終了する（図１２（ｂ）参照）。
【００５１】前半のＰＭＭにおいては、終了を示す値の送出（ステップ１３１６参照）により、および、全ての要素の順位番号の確定（ステップ１３３６でイエス(Yes)）により処理が終了する。上記処理と同様の手順にて、ＰＭＭ１４－３とＰＭＭ１４－４との間でも、マージ処理が実行され、これにより、図１２（ｃ）に示すように各要素の順位番号が確定する。
【００５２】２つのＰＭＭにおける各要素の順序番号が確定すると、ＣＰＵ１２は、スイッチを切り換えて、各々が２つのＰＭＭからなる、２つのＰＭＭ群の間を接続する。図１４（ａ）および図１４（ｂ）は、それぞれ、図８に示すＰＭＭにおける、２つのＰＭＭ群の接続の一例を示す図である。図１４（ａ）においては、ＰＭＭ１４－１、１４－２が、第１のＰＭＭ群を構成し、ＣＰＵ１２は、ＰＭＭ１４－３、１４－４が第２のＰＭＭ群を構成し、ＰＭＭ１４－１、１４－２の出力と、ＰＭＭ群１４－３の入力とが接続され、ＰＭＭ１４－３の出力とＰＭＭ１４－４の入力とが接続され、かつ、ＰＭＭ１４－４の出力とＰＭＭ１４－１、１４－２の入力とが接続されるように、スイッチ２８、３０を制御する（図７のステップ７０９参照）。或いは、図１４（ｂ）に示すように、ＰＭＭ１４－１、１４－２の出力が、ＰＭＭ１４－３、１４－４と接続されるようにスイッチが制御されても良い。
【００５３】図１５（ａ）、（ｂ）は、それぞれ、図１４（ａ）、（ｂ）を模式的に表わした図である。後に明らかになるが、図１５（ａ）において、ＰＭＭ１４－４からＰＭＭ１４－１、１４－２に与えられるデータ（図中、符号①参照）は順位番号を示し、ＰＭＭ１４－１、１４－２からＰＭＭ１４－３に与えられるデータ（図中、符号②参照）は要素を示し、かつ、ＰＭＭ１４－３からＰＭＭ１４－４に与えられるデータ（図中、符号③参照）は、要素およびＰＭＭ１４－３が算出した仮想順位番号を示す。また、図１５（ｂ）においても、各ＰＭＭ間において授受されるデータ①、②は、図１５（ａ）のものと同じであり、その一方、ＰＭＭ１４－３からＰＭＭ１４－４に伝達されるデータ（符号③参照）は、ＰＭＭ１４－３が算出した仮想順位番号を示す。
【００５４】上記図１２に示すように、２つのＰＭＭのペアにおける配列の部分およびこれらに含まれる要素の順位番号から、２つのＰＭＭ群におけるマージ処理および配列の順位番号を決定する処理（図７のステップ７０９参照）につき、説明を加える。なお、以下の説明では、図１４（ｂ）、図１５（ｂ）に示すバスの接続態様にしたがって、各ＰＭＭにて実行する処理を説明する。
【００５５】まず、各々がＰＭＭ１４－１およびＰＭＭ１４－２（以下、「前半のＰＭＭ群」と称する。）において、ＰＵＴポインタを初期位置に配置する（ステップ１３０１）。なお、以後の処理では、ＰＵＴポインタは、前半のＰＰＭ群を構成するＰＭＭにおいて、自己の掌握する要素が送出されるのにしたがって移動する。その一方、後半のＰＭＭの各々は、比較ポインタを、その構造体配列を初期化するとともに、初期位置に配置する（ステップ１３０２）。次いで、前半のＰＭＭ群を構成する各ＰＰＭにおいては、現在、前半のＰＭＭ群を構成する各ＰＭＭは、どの順位番号の要素が送出されたかを把握している。なお、フローチャートでは、ＰＵＴポインタとして、送信時に利用する送信ポインタと受信ポインタと双方を用いているが、基本的にこれら送信ポインタおよび受信ポインタの移動は、後半のＰＭＭ群における処理時間を挟むが、僅かな時間差のみをもって行われている。たとえば、後述するように、あるＰＭＭにおいて送信ポインタをインクリメントとした（ステップ１３０４参照）場合には、当該ＰＭＭは、受信処理においても、受信ポインタをインクリメントする（ステップ１３３５参照）。
【００５６】前半のＰＰＭ群を構成する各ＰＭＭは、処理の対象となる要素の順位番号に基づき、当該要素が、自己が掌握しているものか否かを判断する（ステップ１３０２）。このステップ１３０２にてイエス(Yes)と判断された場合には、ＰＵＴポインタの指し示す要素を、バス２４を介して、ＰＭＭ１４－３、１４－４に伝達する（図１３のステップ１３０２、および、図１６（ａ）参照）。上記例では、まず、順位番号「０」である要素「あべ」が、ＰＭＭ１４－１から、ＰＭＭ１４－３およびＰＭＭ１４－４に伝達される。この処理により、ＰＭ

Ｍ１４－１においては、ＰＵＴポインタの位置が移動する（ステップ１３０４）。

【００５７】ＰＭＭ１４－３およびＰＭＭ１４－４は、それぞれ、要素を受信し（ステップ１３１２）、その要素を挿入すべき位置を捜し出し（ステップ１３１３）、範囲「１」に属する要素が存在するか否かを判断する（ステップ１３１４）。上記要素「あべ」に関しては、ステップ１３１４においてノー(No)と判断される。これにより、ＰＭＭ１４－３において、要素「あべ」の仮想順位番号は「０」となるため、この値を、ＰＭＭ１４－４に伝達する。ＰＭＭ１４－４においても、要素「あべ」の仮想順位番号は「０」となる。そこで、ＰＭＭ１４－４のＭＰＵは、「MAX（０、０）＝０」を、要素「あべ」の順位番号として、前方のＰＭＭ群にバスを介して返送する（図１６（ｂ）およびステップ１３１５参照）。次いで、ＰＭＭ１４－３、１４－４においては、構造体配列中の前方挿入数（Ｙ）および仮想順位番号（Ｚ）が、それぞれインクリメントする（ステップ１３１６）。上記例では、これにより、それぞれの構造体配列が、（０、１、１）、（０、１、１）となる。

【００５８】後半のＰＭＭ群から順位番号が与えられると（ステップ１３３１）、前半のＰＭＭ群を構成する各ＰＭＭは、現在処理中の要素（たとえば、要素「あべ」）が、自己が掌握するものであるかを判断する（ステップ１３３３）。要素「あべ」の順位番号が伝達された場合には、ＰＭＭ１４－１が、上記ステップ１３３３でイエス(Yes)と判断し、その位置の要素に対応する順位番号を、後半のＰＭＭ群から与えられたものに書き換える（図１６（ａ）およびステップ１３３４参照）。同様に、前半のＰＭＭ群は、次の順位番号を付与された要素を、後半のＰＭＭ群に伝達する。上記例では、ＰＭＭ１４－２から、要素「あべ」が伝達され（図１７（ａ）参照）、後半のＰＭＭ群の各々の仮想順位番号のうち大きなもの「MAX（１、１）＝１」が、当該要素「あべ」の順位番号として、前半のＰＭＭ群に伝達される（図１７（ｂ）参照）。また、後半のＰＭＭ群を構成するＰＭＭ１４－３、１４－４において、構造体配列は、それぞれ、（０、２、２）、（０、２、２）となる（図１７（ｂ）参照）。

【００５９】さらに、前半のＰＭＭ群は、次の順位番号を付与された要素を、後半のＰＭＭに伝達する。上記例では、ＰＭＭ１４－２から、要素「はら」が伝達される（図１８（ａ）参照）。ＰＭＭ１４－３において、要素「はら」は、当該ＰＭＭ１４－３が掌握する要素「たなか」よりも後ろであると判断される（ステップ１３１３参照）。したがって、ＰＭＭ１４－３においては、範囲「１」には、要素「さとう」および要素「たなか」が属するため、要素「さとう」および要素「たなか」の順位番号に、それぞれ、前方挿入数「Ｙ（＝２）」が加えられる。これにより、要素「さとう」の順位番号は「０＋２＝２」、要素「たなか」の順位番号は「２＋２＝４」と決定される（ステップ１３１７参照）。次いで、ＰＭＭ１４－３のＭＰＵは、構造体配列（現在の値は（０、２、２））の仮想順位番号Ｚに、範囲「１」中の末尾の要素の順位番号「４」を与え（ステップ１３１８参照）、未処理位置を進める（すなわち、Ｘの値を「０」から「２」にする）（ステップ１３１９参照）。さらに、ＰＭＭ１４－３のＭＰＵは、構造体配列（現在の値は（２、２、４）の前方挿入数「Ｙ」および仮想順位番号「Ｚ」をインクリメントする（ステップ１３２１参照）。これにより、構造体配列は、（２、３、５）となる。ＰＭＭ１４－３における仮想順位番号「Ｚ（＝５）」は、バスを介してＰＭＭ１４－４に伝達される。その後に、ＰＭＭ１４－３のＭＰＵは、構造体配列の仮想順位番号「Ｚ」をインクリメントする（ステップ１３２２参照）。上記例では、ステップ１３２２を施すことにより、構造体配列は、（２、３、６）となる。

【００６０】その一方、ＰＭＭ１４－４において、要素「はら」は、当該ＰＭＭ１４－４が掌握する要素「すぎもと」と「よしだ」との間に位置すると判断される（ステップ１３１３参照）。したがって、ＰＭＭ１４－４において、範囲「１」には、要素「すぎもと」が属するため、要素「すぎもと」の順位番号に、前方挿入数「Ｙ（＝２）」が加えられ、これにより、要素「すぎもと」の順位番号は、「１＋２＝３」と決定される（ステップ１３１７参照）。次いで、ＰＭＭ１４－４のＭＰＵは、構造体配列（現在の値は（０、２、２））の仮想順位番号「Ｚ」に、範囲「１」中の末尾の要素の順位番号「３」を与え（ステップ１３１８参照）、未処理位置を進める（すなわち、「Ｘ」の値を「０」から「１」にする）（ステップ１３１９参照）。さらに、ＰＭＭ１４－４のＭＰＵは、構造体配列（現在の値は（１、２、３））の前方挿入数「Ｙ」および仮想順位番号「Ｚ」をインクリメントする（ステップ１３２１参照）。これにより、構造体配列は（１、３、４）となる。

【００６１】この後に、ＰＭＭ１４－４は、ＰＭＭ１４－３から与えられた仮想順位番号「Ｚ（＝５）」と、自己が算出した仮想順位番号「Ｚ（＝４）」とを比較し、より大きな方の値である「MAX（５、４）＝５」を、伝達された要素「はら」の順位として、前半のＰＭＭ群に伝達する（ステップ３２１参照）。これにより、前半のＰＭＭ群において（より詳細には、要素「はら」を送出したＰＭＭ１２－２において）、当該要素の順位番号が「５」であることが確定する。なお、ＰＭＭ１４－４においても、ステップ１３２１の後に、構造体配列中の仮想順位番号「Ｚ」がインクリメントされる（ステップ１３２２参照）。上記例では、構造体配列は、（１、３、５）となる。

【００６２】同様に、前半のＰＭＭ群より要素「やまもと」が送出される（図１９（ａ））が、この場合の処理

も、図１３にしたがって実行される。再度、簡単に説明すると、要素「やまもと」を受理したＰＭＭ１４－３においては、要素「やまもと」の挿入位置より前方に、範囲「１」に属する要素が存在しないため、ＰＭＭ１４－３は、その構造体配列中の仮想順位番号「Ｚ（＝６）」を、ＰＭＭ－４に伝達する。ＰＭＭ１４－４においても、要素「やまもと」の挿入位置より前方に、範囲「１」に属する要素が存在しないため、その構造体配列中の仮想順位番号「Ｚ（＝５）」と、伝達された仮想順位番号「Ｚ（＝６）」とを比較して、その大きな方（MAX（６、５）＝６）を、要素「やまもと」の順位番号として、前半のＰＭＭ群に返送する（図１９（ｂ）およびステップ１３１５参照）。前半のＰＭＭ群においては、要素「やまもと」を送出したＰＭＭ１４－１が、要素「やまもと」に対応する順位番号を、受理した順位番号（＝６）に書き換える。なお、ＰＭＭ１４－３においては、ステップ１３１６を経ることにより、その構造体配列は（２、４、７）となり、その一方、ＰＭＭ１４－４においては、ステップ１３１６を経ることにより、その構造体配列は（１、４、６）となる。

【００６３】このようにして、前半のＰＭＭ群において全ての要素の送出が終了すると、前半のＰＭＭ群を構成する、何れかのＰＭＭが、終了を示す値を後半のＰＭＭ群に伝達する（ステップ１３０６参照）。後半のＰＭＭ群を構成する各ＰＭＭは、これを受理して、それぞれ、ステップ１３１２ないしステップ１３２３の処理を実行する。上記例では、ＰＭＭ１３－４においては、順位の確定していない要素「よしだ」が存在する。このため、ＰＭＭ１３－４においては、ステップ１３１４においてイエス(Yes)と判断され、範囲「１」に属する要素「よしだ」の順位番号に、前方挿入数「Ｙ」を加えて「３＋４＝７」、得られた数「７」を、要素「よしだ」の順位番号とする。このような処理を経た後、後半のＰＭＭ群を構成する各ＰＭＭにおいて、ステップ１３２３にてイエス(Yes)と判断され、後半のＰＭＭ群における処理も終了する。

【００６４】上記例では、４つのＰＭＭに配列中の要素が格納されていたが、それ以上のＰＭＭ中に配列中の要素が格納されている場合には、さらに、４つのＰＭＭを一群のＰＭＭとして、各々が４つのＰＭＭからなるＰＭＭ群のペアを作成し、これらペアの間で、略同様の処理を実行すれば良い。たとえば、図２０に示すように、１０２４個のＰＭＭにて、ある配列中の要素が格納されていると考える。この場合には、まず、ＰＭＭ１およびＰＭＭ２、ＰＭＭ３およびＰＭＭ４、ＰＭＭ５およびＰＭＭ６、…ＰＭＭ１０２３およびＰＭＭ１０２４を、それぞれ連結し（ＰＭＭ間の実線参照）、これら２つのＰＭＭ間で、要素の順位番号を確定し、次いで、ＰＭＭ１およびＰＭＭ２を前半のＰＭＭ群、ＰＭＭ３およびＰＭＭ４を後半のＰＭＭ群とするＰＭＭ群のペア、ＰＭＭ５およびＰＭＭ６を前半のＰＭＭ群、ＰＭＭ７およびＰＭＭ８（図示せず）を後半のＰＭＭ群とするＰＭＭ群のペア、…ＰＭＭ１０２１およびＰＭＭ１０２２（図示せず）を前半のＰＭＭ群、ＰＭＭ１０２３およびＰＭＭ１０２４を後半のＰＭＭ群とするＰＭＭ群のペアを形成して、各ペア間を連結し（破線参照）、これらペアを構成する２つのＰＭＭ群の間で、要素の順位番号を確定する。以下、４つのＰＭＭを前半のＰＭＭ群、および、これに引き続く４つのＰＭＭを後半のＰＭＭ群とするＰＭＭ群のペア（一点鎖線参照）、８つのＰＭＭを前半のＰＭＭ群、および、これに引き続く８つのＰＭＭを後半のＰＭＭ群とするＰＭＭ群のペア（点線参照）というように、各々が2^nのＰＭＭ群からなるＰＭＭ群のペアを順次形成し、これらの間で、要素の順位番号を確定する。最終的に、５１２個のＰＭＭを前半のＰＭＭ群、それに引き続く５１２個のＰＭＭを後半のＰＭＭ群とするＰＭＭ群のペアの間で、要素の順位番号を確定することにより、１０２４個のＰＭＭ中の要素すべての順位番号を確定することが可能となる。

【００６５】このように、各々が2^n個のＰＭＭからなるＰＭＭ群のペアを形成して、ペアを構成するＰＭＭ群の各ＰＭＭに格納された要素の順位番号を順次確定することにより（図７のステップ７０９、７１０参照）、最終的に、全ての要素の順位番号が確定すると（ステップ７１０でイエス(Yes)、必要な場合には、上記順位番号付けにしたがった配列を再形成する処理が実行される（ステップ７１１）。この処理は必須ではないが、順位番号にしたがって要素が配置されているような配列を生成することにより、後に実行される情報処理をより高速に実現することが可能となる。

【００６６】より詳細には、まず、ＣＰＵ１２は、各ＰＭＭの入力および出力があるバスに接続されるように、スイッチ２８、３０を制御する。図２１は、ＰＭＭが４つである場合に、これらの間の接続を模式的に示す図である。次いで、ＰＭＭ１４－１～１４－４のＭＰＵは、確定した順位番号にしたがって、要素および順位番号をバス上に放出する。各ＭＰＵは、バス上に放出される要素およびその順位番号をモニターし、もともと自己のＲＡＭコアにて分掌していた要素の添え字（レコード番号）と同一の順位番号を有する要素を取り込み、ＲＡＭコアの所定の領域に格納する。たとえば、添え字（レコード番号）「０」および「１」の要素をもともと自己のＲＡＭコアに記憶していたＰＭＭにおいては（たとえば、図１０のＰＭＭ１４－１参照）、順位番号「０」および「１」を付された要素を取り込み、これらを記憶すれば良い。このようにすれば、各ＰＭＭにおいて、実際にソートされた配列を分掌することが可能となる。なお、このように、ソートされた配列を形成する際にも、ＰＭＭのＭＰＵは、必要な空間ＩＤ管理テーブルを作成する。

【００６７】或いは、図２２に示すように、ソートされた配列を分掌するための他のＰＭＭ（ＰＭＭ１４－５～ＰＭＭ１４－８）を設け、他のＰＭＭ群の各々が、ＰＭＭ１４－１～ＰＭＭ１４－４から、順次出力される要素およびその順位番号をモニターし、順位番号にしたがって、自己が取り込むべき要素を取り込んで、各ＰＭＭのＲＡＭコアに記憶しても良い。たとえば、上記本発明を利用して、１０２４個のＰＭＭを設け、各ＰＭＭに約１００万のデータ（要素）を格納しておき、これらデータのソートを行った場合に、以下のような時間でソートが完了すると考えられる。ここに、各ＰＭＭ間を接続するバスは、６．４ＧＢ／秒のデータ伝送が可能であり、かつ、処理中には、全てのＰＭＭが並列的に動作し（すなわち、処理を実行していないＰＭＭが存在せず）、かつ、関連するすべてのＰＭＭが、同時にかつ協調的に同動作できると仮定する。また、各ＰＭＭにおける約１００万のデータ（要素）のソートは、２．５秒にて完了すると考える。この場合、１０２４個のＰＭＭ中の、約１０億個の要素をソートするために、略４秒程度しか必要としないことがわかる。

【００６８】本実施の形態によれば、各ＰＭＭを、初期的には、２つのＰＭＭのペアに分け、次いで、順次、各群が2^n個のＰＭＭから構成されるＰＭＭ群のペアに分けて、各ペアの間で順位番号を確定させていく。また、各ペアにて利用するバスを、スイッチ等を用いて調整することにより、各ペアにおける順位番号の確定が、並列的に実行することができる。さらに、前半のＰＭＭ群からの要素を、後半のＰＭＭ群に伝達し、後半のＰＭＭ群における構造体配列中の値にしたがって確定させ、確定された順位を、前半のＰＭＭ群に伝達する手順を繰り返すことにより、各ペアにおける順位番号の確定をなすことができる。したがって、処理を実行していない（いわゆる「遊んでいる」）ＰＭＭが生じることなく、極めて並列的に処理を実行できるとともに、バスを利用したデータ転送量を減じることができる。これにより、ソート速度を著しく高速にすることが可能となる。

【００６９】なお、上記第１の実施の形態においては、図１４（ｂ）および図１５（ｂ）に示すようにＰＭＭを接続し、これらの間で各要素に順位番号を付することにソート処理を実現しているが、図１４（ａ）および図１５（ａ）に示すようにＰＭＭを接続しても良い。この場合には、図１３における後半のＰＭＭ群の処理（ステップ１３１２～ステップ１３２３）が並列的に実行されず、あるＰＭＭにおいて仮想順次番号が得られると、処理対象となる要素と当該仮想順位番号が、隣接するＰＭＭに伝達され、当該ＰＭＭにおいて、ステップ１３１２～１３２３の処理が実行される。したがって、後半のＰＭＭ群を構成するＰＭＭの個数が多くなると、それだけ処理の遅延を招く場合もある。

【００７０】［他のソート処理（第２の実施の形態）］

次に、本発明の第２の実施の形態につき説明を加える。上記第１の実施の形態においては、すべての要素（前半のＰＭＭ群内の要素）が、後半のＰＭＭ群に転送されている。しかしながら、配列が巨大になるにしたがって、重複値が多数出現し得る。上記第１の実施の形態にかかる手法では、同じ値をとる要素が何度もバス上に送出される。場合によっては、同じ値の要素を繰り返し送出することは、無駄であると考えることができる。そこで、第２の実施の形態においては、ＰＭＭ群中の要素の個数を予めカウントし、要素とともにその個数を、後半のＰＭＭ群に送出することにより、重複する要素を繰り返しバス上に送出することを防止している。

【００７１】たとえば、４つのＰＭＭの対の各々においてソート処理が終了し、これら対を接続して、８つのＰＭＭにおけるソート処理を実行することを考える。この場合に、図２３に示すように、８つのＰＭＭのマージおよびソート処理を実行するバス（図２３においてＰＭＭの下側に位置するバス、たとえば、符号２３０１～２３０３参照）のほか、他のバス（図２３においてＰＭＭの上側に位置するバス２３０４、２３０５参照）を用いて、ＰＭＭ間のデータの授受ができるのが望ましい。図２３に示すような接続態様において、ＰＭＭ１４－１～ＰＭＭ１４－４（以下、便宜上、「ＰＰＭ１４－１」ないし「ＰＭＭ１４－４」を、それぞれ、「ＰＭＭ１」ないし「ＰＭＭ４」と称する。）における値の重複数を算出する処理につき説明を加える。ここに、ＰＭＭ１～ＰＭＭ４の入出力端子（Ｉ／Ｏ）と接続されたバス（符号２３０４参照）を第１のバスと称し、ＰＭ１～ＰＭＭ４の他の入出力端子（Ｉ／Ｏ）と接続されたバス（符号２３０５参照）を第２のバスと称する。第１のバスは、ＰＭＭ１～ＰＭＭ４からなるＰＭＭ群の情報交換用に用いられ、第２のバスは、値およびその重複数を各ＰＭＭに与えるために利用される。

【００７２】なお、以下の説明においては、図２５に示すように、ＰＭＭ１～ＰＭＭ４中の配列において、各要素に順位番号が付されたものの重複数を算出している。すなわち、重複数は、前半のＰＭＭ群においてのみ算出すれば足りる。図２４は、ＰＭＭ群における重複数を算出するための処理を示すフローチャートである。ＰＭＭ１～ＰＭＭ４の各々は、まず、種々の初期化の処理を実行する（ステップ２４０１）。ここで、各ＰＭＭでは、処理にかかる値（要素）の順位番号を示す順位番号カウンタ、ある値（要素）がどれだけ重複して存在するかを示す同一値個数カウンタ、および、前回の処理において処理対象となった値（要素）を保持する前回値保存レジスタが設けられ、順位番号カウンタおよび同一値個数カウンタの値に初期値「０」が与えられる（図２５参照）。なお、初期的には、前回値保存レジスタには何も値が保持されない。

【００７３】次いで、各ＰＭＭは、順位番号カウンタを

参照して、処理対象となる要素の順位番号を特定し、当該順位番号を付された要素が、自己の掌握するものであるか否かを判断する（ステップ２４０３）。上記例では、初期的に順位番号カウンタのカウンタ値は「０」であるため、ＰＭＭ３が、自己の掌握する要素が処理対象であると判断する（ステップ２４０３でイエス(Yes)）。なお、次のステップ２４０４～２４０５は、最初の処理（すなわち順位番号「０」の要素に関する処理）では無視される。ＰＭＭ３は、順位番号「０」が付された要素（この場合には「あべ」）と同じ要素が、自分の中にいくつ存在するか（すなわち、ＰＭＭ３がいくつ「あべ」という要素を掌握しているか）を判定し、第１のバスに、要素「あべ」と、この要素をいくつ持っているかを示す自己ＰＭＭ内存在数とを送出する（ステップ２４０６）。他のＰＭＭ（ＰＭＭ１、ＰＭＭ２およびＰＭＭ４）では、ステップ２４０３でノー(No)と判断されるため、ステップ２４０７に進む。

【００７４】各ＰＭＭは、第１のバスを介して与えられたデータを受理し、データ中の自己ＰＭＭ内存在数に基づき、順位番号カウンタのカウンタ値に、ＰＭＭ内存在数を加える（ステップ２４０８）。上記例では、順位番号カウンタのカウンタ値が「０＋１＝１」となる。次いで、与えられた要素が、前回値保存レジスタのものと異なるか否かが判断され（ステップ２４０９）、双方が同一であった場合には、同一値個数カウンタのカウンタ値に、自己ＰＭＭ内存在数が加えられ（ステップ２４１０）、その一方、新しい値のときには、後述する入替処理が実行される（ステップ２４１１）。なお、初回の処理では、前回値保存レジスタには何ら値が保持されていないため、上記ステップ２４０９の判断が省略され、かつ、前回値保存レジスタに、要素が収容されるとともに、同一値個数カウンタのカウントアップが実行される。したがって、上記例においては、各ＰＭＭは、受理した要素「あべ」を前回値保存レジスタに記憶するとともに、同一値個数カウンタを「０＋１＝１」とする（図２６参照）。このようなステップ２４０１～２４１１の処理が繰り返され、最後の要素に関する処理が終了すると、ステップ２４０１においてイエス(Yes)と判断され、ステップ２４１２に進む。

【００７５】上記例において、最初のステップ２４０１～２４１１の処理が終了すると、各ＰＭＭは、順位番号カウンタのカウンタ値を参照して、カウンタ値が「１」であることを確認する。これにより、ＰＭＭ４が、順位番号「１」の要素を掌握していることがわかる。また、ＰＭＭ４は、前回値保存レジスタの値（要素「あべ」）と、順位番号「１」が付された要素「あべ」とを比較し（ステップ２４０４）、値に変化がないため、要素「あべ」および自ＰＭＭ内存在数「１」を第１のバスに送出する（ステップ２４０５）。第１のバスを介してデータを受理した各ＰＭＭは、図２７に示すように、順位番号

カウンタをカウントアップ（１＋１＝２）し（ステップ２４０８）、また、前回値保存レジスタに記憶された値と、受理した要素とが同一であるため、同一値個数カウンタをカウントアップ（１＋１＝２）する（ステップ２４１０）。

【００７６】この後に、各ＰＭＭにおいては、順位番号「２」の要素に関する処理が実行される。順位番号「２」の要素の処理では、ＰＭＭ１が要素を保持しているため、ＰＭＭ１が、要素「いとう」と前回値保存レジスタに記憶された要素「あべ」とを比較する。ここでは、値に変化があるため（ステップ２４０４でイエス(Yes)）、ＰＭＭ１は、前回値保存レジスタの内容（要素「あべ」）と、同一値個数カウンタの値「２」を第２のバスに送出する（ステップ２４０５）。このレジスタの内容およびカウンタ値は、各ＰＭＭに与えられる。後述するように、ある要素（この場合には要素「あべ」）の重複数が算出されると、当該要素に関するソート処理（図３１参照）が実行され得る。したがって、各ＰＭＭにおいて、要素およびその重複数は、当該要素に関するソート処理が完了するまで保持していれば良い。また、要素「いとう」および自ＰＭＭ内存在数「１」が第１のバスに与えられる（ステップ２４０６）。

【００７７】各ＰＭＭは、第１のバスを介して与えられたデータに基づき、順位番号カウンタをカウントアップ（２＋１＝３）する（ステップ２４０８）。また、前回値保存レジスタの値「あべ」と、伝達された要素「いとう」とが異なるため（ステップ２４０９でイエス(Yes)）、各ＰＭＭは、前回値保存レジスタの値を書き替える（更新する）とともに、同一値個数カウンタの値を、第１のバスを介して与えられた自ＰＭＭ内存在数に置き換える（ステップ２４１１および図２８（ａ）参照）。

【００７８】他の順位番号の要素についても、同様の処理が施される。たとえば、順位番号「３」に関しては、ＰＭＭ３が、ステップ２４０４、２４０６にしたがって、要素「いとう」を第１のバスに送出し、かつ、各ＰＭＭが、ステップ２４０７、２４０８、２４０９および２４１０にしたがって、各カウンタをカウントアップする（図２８（ｂ）参照）。また、順位番号「４」に関して、ＰＭＭ１が、ステップ２４０４、２４０５、２４０６にしたがって、要素「いとう」および同一値個数カウンタのカウンタ値「２」を第２のバスに送出するとともに、要素「すぎもと」を第１のバスに送出し、かつ、各ＰＭＭが、ステップ２４０７、２４０８、２４０９、２４１１の順で、各カウンタをカウントアップするとともに、レジスタを更新する（図２９（ａ）参照）。

【００７９】順位番号「５」に関する処理では、ＰＭＭ２は、自己が掌握する要素「すぎもと」が２つあることから、第１のバスに、要素「すぎもと」および自ＰＭＭ内存在数「２」を送出する。したがって、各ＰＭＭにおいて、順位番号カウンタ、および、同一値個数カウンタ

のそれぞれのカウンタ値に、「２」が加えられる（図２９（ｂ）参照）。また、この処理により、順位番号カウンタのカウンタ値は、「５」から「７」に変化するため、次の処理対象となる要素の順位番号は「６」ではなく「７」になることに留意されたい。順位番号「７」を付された最後の要素に関する処理（図３０（ａ）参照）が終了すると、ステップ２４０１にてイエス(Yes)と判断される。そこで、先頭のＰＭＭ（上記例では、ＰＭＭ１）は、第２のバスに、要素「すぎもと」および同一値個数カウンタのカウンタ値「４」を送出し（ステップ２４１３）、次いで、第２のバスに処理が終了したことを示すデータを送出する（ステップ２４１４）。各ＰＭＭには、各要素およびその個数を示す存在数が第２のバスを介して与えられ、これがソート処理に利用される。なお、上記例では、先頭のＰＭＭが、ステップ２４１３、２４１４を実行するように構成したが、これに限定されるものではなく、予め、最後の要素等、および、終了を示すデータを出力するＰＭＭを定めておけば良い。上述したように、あるＰＭＭ群における各要素の存在数を得ることにより、ＰＭＭ群を、他のＰＭＭ群とマージして、これらの要素をソートする際に、重複した要素を送る必要がなくなる。

【００８０】図３１は、重複した要素の送出を排除したソート処理を示すフローチャートである。図３１は、一部を除き、図１３の処理と同一であり、その末尾２桁が同じものは、略対応する処理となる。また、図３１において、二重の囲みを付した処理は、新規に追加された処理、或いは、図１３の対応するものと若干異なる処理であることを示している。この処理においては、前半のＰＭＭ群において、処理対象となる要素（すなわち、送出ポインタにより指示される要素）を掌握するＰＭＭは、その要素とともに、前半のＰＭＭ群における当該要素の重複数（存在数）「Ｎ」を、後半のＰＭＭ群に送出する（ステップ３１０３、３１０３－２参照）。たとえば、図２５ないし図３０に示した例において、ＰＭＭ１～ＰＭＭ４からなる前半のＰＭＭ群から、後半のＰＭＭ群に、要素「あべ」が送出される場合には、要素「あべ」のほか、前半のＰＭＭ群における要素「あべ」の重複数「２」が伝達される。また、前半のＰＭＭ群の送出処理において、要素およびその重複数を出力したＰＭＭは、その出力の後に、自己が把握する当該要素の数だけ送出ポインタを移動させる（ステップ３１０４）。たとえば、図２８（ａ）に示すように、要素「あべ」の重複数は「２」であり、これらがＰＭＭ３およびＰＭＭ４において、一つずつ把握されている。したがって、ＰＭＭ３およびＰＭＭ４において、それぞれ、送出ポインタの位置が１つ下方に移動する。なお、各ＰＭＭにおける送出ポインタの移動量の総和が、当該要素の重複数「Ｎ」と等しくなる。

【００８１】その一方、要素およびその重複数を受信した後半のＰＭＭ群を構成する各ＰＭＭにおいては、図３１のステップ３１１６およびステップ３１２０に示すように、前方挿入数および仮想順位番号に、それぞれ、重複数「Ｎ」が加えられる。これは、自分より前方に位置する（順位が小さな）要素が、「Ｎ」だけ存在することに対応している。

【００８２】さらに、前半のＰＭＭ群を構成するＰＭＭの受信処理においては、前半のＰＭＭによる送出処理において送出された要素（比較対象データ）と、後半のＰＭＭによる処理において送出された順位番号とに基づき、受信した順位番号と、比較対象データの送出時における順位番号との差「Ｍ」が算出される（ステップ３１３２－２）。この差「Ｍ」は、後半のＰＭＭ群において、比較対象となっている要素の前方に位置する（すなわち、当該要素より小さい順位番号を付した）要素の数を示している。したがって、後半のＰＭＭ群を構成する各ＰＭＭは、自己の掌握する要素のうち、当該比較対象となっている要素と同一の要素を特定し（ステップ３１３２－３）、存在する場合には、これら要素の順位番号に、それぞれ「Ｍ」を加算する（ステップ３１３４）。ステップ３１３４の後に、ＰＭＭは、当該要素の数だけ受信ポインタを移動させる（ステップ３１３５）。この処理は、ステップ３１０４と略同様である。

【００８３】次に、図２４に示す重複数の算出と、図３１に示すソート処理（場合により、「ソート本体」と称する。）との並列性につき説明を加える。図２３に示すように、本実施の形態においては、重複数のカウントにかかるＰＭＭ間の通信を、バス２３０４、２３０５を利用し、ソート本体の実行にかかるＰＭＭ間の通信を、バス２３０１、２３０２、２３０３等を利用している。そこで、ＰＭＭにおいて並列処理が可能であれば、重複数のカウントとソート本体を並列して実行することができる。この場合に、前半のＰＭＭ群において、ある要素に関する重複数の算出が終了すると（たとえば、図２８（ａ）に示すように、要素「あべ」およびその重複数「２」が第２のバスに送出され、前半のＰＭＭ群を構成するＰＭＭ（ＰＭＭ１～ＰＭＭ４）に受理されると、図３１に示すような処理が、重複数の算出された要素に関して実行可能である。すなわち、ある要素の重複数の算出に応答して、当該要素に関するステップ３１０２～ステップ３１０４の処理、ステップ３１１２～ステップ３１２２の処理、および、ステップ３１３２～ステップ３１３５の処理を実行することができる。また、ある要素に関する要素とその重複数などは、上記図３１に示す処理のうち、当該要素に関するものの終了とともに、削除することが可能である。したがって、前半のＰＭＭ群を構成するＰＭＭの各々において、要素およびその重複数に関するデータ（その量は、異なる要素の数が多くなるに伴って大きくなる）を全て保持しておく必要もない。

【００８４】このように、第２の実施の形態において

は、前半のＰＭＭ群において、重複数を算出し、要素およびその重複数を後半のＰＭＭ群に送出している。したがって、前半のＰＭＭ群が、後半のＰＭＭ群に同一の要素を重複して送る必要がなくなる。特に、同じ要素が数多く重複する場合（たとえば、要素が男女の種別を示すもの、年齢を示すものなど）には、ソート本体の処理回数を減少させることが可能となり、より高速にソート処理を実現することができる。

【００８５】［コンパイル処理（第３の実施の形態）］
次に、本発明の第３の実施の形態につき説明を加える。第３の実施の形態では、各ＰＭＭ内に配置された要素からなる配列に基づいて、レコード、各要素を重複なく配置した値リスト、および、レコードから値リストを指定するためのポインタ配列を作成する。この処理を、本明細書においてコンパイルと称する。たとえば、４つのＰＭＭ（ＰＭＭ１～ＰＭＭ４）に、ある配列の要素が分掌されている場合には、図３２に示すように、ＰＭＭを接続すれば良い。図３２に示すように、ＰＭＭ１～ＰＭＭ４の入出力端子（Ｉ／Ｏ）は、第１のバス（符号３２０１参照）により接続され、その一方、ＰＭ１～ＰＭＭ４の出力端子（Ｏ）および他のＰＭＭ“ｋ”の入力端子（Ｉ）は、第２のバス（符号３２０２参照）により接続されている。

【００８６】第１のバスは、ＰＭＭ１～ＰＭＭ４からなるＰＭＭ群の情報交換用に用いられ、第２のバスは、要素およびその重複数を他のＰＭＭ“ｋ”に与えるために利用される。本実施の形態においては、上記要素およびその重複数に基づき、他のＰＭＭ“ｋ”において、値リストおよび存在数配列等が形成される。なお、このＰＭＭ“ｋ”は、ＰＭＭ１～ＰＭＭ４以外のＰＭＭであっても良いが、無論ＰＭＭ１～ＰＭＭ４の何れかであっても良い。図３３は、本実施の形態にかかるコンパイル処理を示すフローチャートである。なお、説明を容易にするために、図３４（ａ）に示すように、ＰＭＭ１～ＰＭＭ４には、要素が分掌されており、これらの間で順位番号を付す処理が既に実行されていると考える。まず、各ＰＭＭにおいて、処理にかかる値（要素）の順位番号を示す順位番号カウンタ、処理の後の当該値（要素）の順位番号を示す値番号カウンタ、当該要素がどれだけ重複して存在するかを示す同一値個数カウンタ、および、前回の処理において処理対象となった値（要素）を保持する前回値保存レジスタが設けられ、各カウンタに初期値「０」が与えられる（ステップ３３０１および図３４（ａ）参照）。なお、初期的には、前回値保存レジスタには値が保持されない。

【００８７】以下、図３３のステップ３３０２～ステップ３３０６の処理は、図２４のステップ２４０２１～２４０６と略同様である。すなわち、各ＰＭＭは、順位番号カウンタを参照して、処理対象となるようその順位番号を特定し、当該順位番号が付された要素が、自己の掌握するものであるか否かを判断する(ステップ３３０３)。図３４の状態では、順位番号カウンタのカウンタ値が「０」であるため、ＰＭＭ３が、第２のバスに、当該ＰＭＭ３が順位番号「０」を付された要素「あべ」をいくつ保持しているかを示す自己ＰＭＭ内存在数（この場合には「１」）を創出する（ステップ３３０６および（図３４（ｂ）参照）。次いで、ＰＭＭ３は、前回値保存レジスタに記憶された要素と、第1のバスに放出した要素とが比較され、これらが相違する場合には、値番号カウンタのカウンタ値を、第1のバスに送出したようその順位番号に代入する（ステップ３３０７）。なお、図３４の状態では、値番号カウンタのカウンタ値が初期値「０」であるため、要素「あべ」にかかる順位番号は変化しない（図３４（ｂ）参照）。

【００８８】次いで、各ＰＭＭにおいては、第1のバスを介して与えられたデータを受理する（ステップ３３０８）。ステップ３３０８～３３１１の処理は、図２４におけるステップ２４０８～２４０１の処理と略同様である。すなわち、各ＰＭＭは、順位番号カウンタのカウンタ値に、与えられたデータのＰＭＭ内存在数を加え、さらに、与えられたデータのうち、要素が新たなものではない場合（ステップ３３１０でノー(No)）には、同一値個数カウンタのカウンタ値に、ＰＭＭ内存在数を加える（ステップ３３１１および図３４（ｂ）参照）。図３４に示すように、順位番号「０」が付された要素「あべ」に関する処理が終了すると、順位番号「１」が付された要素に関する処理が、同様に実行される（図３５（ａ）参照）。

【００８９】さらに、順位番号「２」が付された要素に関する処理が実行される。ここでは、ＰＭＭ１が、前回値保存レジスタに記憶された要素「あべ」と、順位番号「２」が付された要素「いとう」とを比較する。ここでは、これらが相違しているため（ステップ３３０４でイエス(Yes)）、ＰＭＭ１は、第２のバスに、前回値保存レジスタに記憶された要素と、同一値個数カウンタのカウンタ値とを送出する（ステップ３０５）。次いで、ＰＭＭ１は、第1のバスに、処理対象の要素「いとう」と、ＰＭＭ１が掌握する要素「いとう」の数である自ＰＭＭ内存在数「１」とを送出する（ステップ３３０６）。その後に、ＰＭＭ１は、前回値保存レジスタに記憶された要素と、第1のバスに放出した要素とを比較する。要素「いとう」を放出する場合には、これらが相違するため、要素「いとう」の順位番号に、値番号カウンタのカウンタ値に「１」を加えた値（０＋１＝１）を代入する。各ＰＭＭは、第1のバスを介して与えられたデータを受理し（ステップ３３０８）、順位番号カウンタのカウンタ値に、受け入れたデータ中の自ＰＭＭ内存在数を加算する（２＋１＝３）（ステップ３３０９および図３５（ｂ）参照）。要素「いとう」が与えられた場合には、前回値保存レジスタの要素「あべ」と与えられた

要素「いとう」とが異なるため（ステップ３３１０でイエス(Yes)）、各ＰＭＭは新値登録処理を実行する（ステップ３３１２）。この処理においては、値番号カウンタのカウンタ値がインクリメント（０＋１＝１）され、同一値個数カウンタのカウンタ値が、受理したデータ中の自ＰＭＭ内存在数「１」に変更され、かつ、前回値保存レジスタの内容が、要素「いとう」に書きかえられる（図３５（ｂ）参照）。
【００９０】順位番号「３」の要素「いとう」に関しても同様の処理が施される。たとえば、ＰＭＭ３は、第1のバスに要素「いとう」および自ＰＭＭ内存在数「１」を送出し（ステップ３３０６参照）、かつ、当該要素「いとう」の順位番号に、値番号カウンタのカウンタ値「１」を代入する（ステップ３３０７および図３６（ａ）参照）。また、各ＰＭＭは、順位番号カウンタのカウンタ値に、受理した自ＰＭＭ内存在数「１」を加える（ステップ３３０９参照）とともに、同一値個数カウンタのカウンタ値に、自ＰＭＭ内存在数「１」を加える（ステップ３３１１および図３６（ａ）参照）。さらに、順位番号「４」の要素「すぎもと」に関しても、図３６（ｂ）に示すように、ＰＭＭ１が、第２のバスに、要素「いとう」および同一値個数カウンタのカウンタ値「２」を送出し（ステップ３３０５）、第1のバスに、要素「すぎもと」および自ＰＭＭ内存在数「１」を送出し（ステップ３３０６）、かつ、要素「すぎもと」の順位番号に、値番号カウンタに「１」を加えた値（１＋１＝２）を代入する。その一方、各ＰＭＭにおいても、順位番号カウンタのカウントアップおよび新値登録処理が実行される（ステップ３３０９、３３１２および図３６（ｂ）参照）。他の順位番号の要素についても、同様の処理が施される。各要素についての処理は、図３７（ａ）、（ｂ）および図３８に示されている。なお、図３８に関して、ＰＭＭ１は、最後の要素「すぎもと」、および、その要素の存在数を第1のバスに送出し、かつ、終了を示すデータを第２のバスに出力する（ステップ３３１５参照）。
【００９１】前述したように、第２のバスには、ＰＭＭ“ｋ”の入力が接続されている。したがって、第２のバスには、重複のない要素と、これに関する値番号カウンタのカウンタ値とが与えられる。したがって、ＰＭＭ“ｋ”はこれらを受理し、受理した要素を、値リストに順次配置するとともに、受理した値番号カウンタのカウント値を、存在数配列に順次配置する。図３９（ａ）は、ＰＭＭ“ｋ”内に作成された値リストおよび存在数配列を示す図である。これらは、ステップ３３０５或いはステップ３３１４にて送出され（図３５（ｂ）、図３６（ｂ）および図３８参照）、ＰＭＭ“ｋ”に伝達されている。図３９（ａ）に示すように、要素が重複なく値リストに配置され、かつ、各要素がいくつ存在するかを示す存在数（すなわち重複数）が存在数配列に配置されている。
【００９２】さらに、ＰＭＭ１～ＰＭＭ４において、レコードと、各要素に、重複がないように付された順位番号とを対応させる値リストへのポインタ配列を作成することができる。つまり、レコードと、当該レコードに対応する要素に付された順位番号とを対応させた配列を作成すれば、これが値リストへのポインタ配列とすることができる（図３９（ｂ）参照）。図３９（ｂ）において、レコード「０」に関して、対応する要素の順位番号「２」が、値リストへのポインタ配列におけるポインタ値となる。これは、値リスト（図３９（ａ）参照）において、格納位置番号「２」であるような値を指示すべきことを示している。すなわち、値リストへのポインタ配列のポインタ値により、ＰＭＭ“ｋ”に格納された値リストを指示することができ、これにより、レコードから要素を特定することが可能となる。
【００９３】このように本実施の形態によれば、ＰＭＭに分掌された配列の要素をソートして順位番号を付し、かつ、同一の要素には同一の順位を付すように、順位を振り直している。要素は、新たに得られた重複のない順位と対応付けられて、値リストに格納される。当該順位は、値リストへのポインタ配列として、分掌された配列中の要素に対応つけられている。したがって、レコードに基づき、ポインタ配列のポインタ値を経て、値リスト中の要素を特定することが可能となる。
【００９４】［値リストの共有化（第４の実施の形態）］次に、本発明の第４の実施の形態につき説明を加える。第４の実施の形態においては、二つの配列を共有化（ジョイン）している。この前提として、コンパイル処理による値リスト、および、値リストへのポインタ配列が作成されている。また、値リストや、値リストへのポインタ配列には、空間ＩＤが付され、各ＰＭＭは、当該空間ＩＤ等により、自己が分掌している配列に関する種々の情報を把握している。
【００９５】図４０は、第４の実施の形態にかかる共有化処理を示すフローチャートである。説明を容易にするために、図４１（ａ）に示すように、元のデータとして、レコードに対応した要素からなる配列（符号４１００参照）が、あるＰＭＭ群に分掌されていると考える。このレコード群に関するコンパイル処理により、ＰＭＭ１およびＰＭＭ２からなるＰＭＭ群に、ポインタ配列（符号４１０１参照）および値リスト（符号４１０２参照）からなるブロック（以下、「情報ブロック」と称する。）が形成されている。その一方、元のデータとして、レコードに対応する要素からなる他の配列（符号４１１０参照）が、他のＰＭＭ群に分掌され、かつ、コンパイル処理により、ＰＭＭ群３およびＰＭＭ群４からなるＰＭＭ群に、ポインタ配列（符号４１１１参照）および値リスト（符号４１１２参照）からなる情報ブロックが形成されていると考える。

【００９６】各PMMに、ＣＰＵ１２から、値リストのジョインを指示するインストラクションが、二つの値リストを示す配列の空間ＩＤとともに伝達される。各ＰＭＭのうち、ジョインすべき配列が自己の掌握する値リスト或いはその部分であるようなもの（つまり、上記例では、ＰＭＭ１～ＰＭＭ４）は、空間ＩＤに基づき、ジョインの対象となる値リストを特定する（ステップ４００１および図４２（ａ）参照）。次いで、ＰＭＭ１～ＰＭＭ４は、二つの値リストを合併した状態で、これらをソートして各要素に順位番号を付与する（ステップ４００２）。このソート処理のために、第１の実施の形態にかかるソート処理を利用することができる。上記例においては、まず、ＰＭＭ１およびＰＭＭ２からなる第１のＰＭＭ群、および、ＰＭＭ３およびＰＭＭ４からなる第２のＰＭＭ群のそれぞれにおいて、要素の順位番号付与の処理を実行し、次いで、第１のＰＭＭ群を前半のＰＭＭ群とし、かつ、第２のＰＭＭ群を後半のＰＭＭ群とすることで、二つのＰＭＭ群中の要素に順位番号を付与する。図４２（ｂ）は、このようにして要素に順位番号が付された状態を示す図である。

【００９７】その後に、処理対象となる値リストを分掌するＰＭＭの間でコンパイル処理が実行され、これにより、他のＰＭＭ或いはＰＭＭ１～ＰＭＭ４の何れかに、共通化された値リストおよび共通化された存在数配列が生成される（ステップ４００３）。すなわち、コンパイル処理により、マージされた値リストの要素が重複しないような新たな値リストと、各要素がどれだけ重複して存在しているかを示す存在数を格納した存在数配列が得られる（図４２（ｃ）参照）。このような処理の後に、ジョインされた新たな値リスト（すなわちコンパイル処理により得られた値リスト）を指示するための新たなポインタ配列が求められる。これは、共有化前の情報ブロックにおけるポインタ配列中のポインタ値が示す、コンパイル処理により得られた順位番号配列の対応する順位番号を、当該ポインタ配列中のポインタ値の位置と対応する位置に格納するような新たなポインタ配列を作成することにより実現される。上記順位番号配列中の値は、各要素に付された新たな順位番号（図４２（ｃ）参照）に対応することは理解できるであろう。

【００９８】図４３（ａ）に示すように、たとえば、ポインタ配列中の第１のポインタ値「１」の示す位置の順位番号配列中の値（順位番号）は「２」であるため、共有化された後のポインタ配列中の対応する位置のポインタ値は「２」となる。また、第２のポインタ値「２」の示す位置の順位番号配列中の値（順位番号）は「３」であるため、共有化されたポインタ配列中の対応する位置のポインタ値は「３」となる。このようにして、ジョインされた値リストに関するポインタ配列を得ることが可能となる（図４３（ａ）および図４３（ｂ））。

【００９９】このような、新たなポインタ配列、および、ジョインされた値リストにより、レコードから値（要素）を特定できることは明らかであろう。図４４に示すように、レコードが、新たに得られた値リストへのポインタ配列中、対応する位置のポインタ値を特定し、かつ、当該ポインタ値が、その値が示す位置にある、値リスト中の要素を特定する。ここで、二つの値リストがジョインされているにもかかわらず、元のデータの要素と同一の要素が指定されることが理解できるであろう。

【０１００】このように、第４の実施の形態によれば、複数の値リストを併合して、併合された値リストの要素に関して、ソート処理とコンパイル処理とを組み合わせることにより、ジョインされた値リスト、および、各値リストの順位番号配列を得る。レコードから値リストを指定するためのポインタ配列により順位番号配列の値（順位番号）が特定され、当該順位番号を、レコードに対応する位置に格納することにより、レコードに基づきジョインされた値リストを指定するための、新たなポインタ配列を得ることができる。したがって、上述したソート処理の時間およびコンパイル処理の時間程度で、複数の値リストをジョインすることが可能となり、著しく処理速度を向上させることが可能となる。

【０１０１】本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。たとえば、前記実施の形態においては、本発明を、コンピュータシステムに適用しているがこれに限定されるものではなく、パーソナルコンピュータなどに接続可能なコンピュータボードに適用することもできる。この場合には、図１において、ＣＰＵ１２、メモリユニット１４、バス２４等がボード上に搭載され、これが、本発明における情報処理ユニットを構成する。

【０１０２】また、ＣＰＵ１２とメモリモジュール１４との間、および／または、メモリモジュール１４間を接続するバスの組の数は、前記実施の形態に限定されるものではなく、コンピュータシステムを搭載する回路基板の大きさ、各バスのビット数などを考慮して適宜決定することができる。また、前記実施の形態においては、メモリモジュールの入出力とバスとの接続を規定するためのスイッチ２８と、ＣＰＵとメモリモジュールとの間、メモリモジュール間、或いは、メモリモジュールの入出力間で、バスの切断することができるスイッチ３０とを設けている。スイッチ２９、３０を設けることにより、たとえば、あるバス（図１のバス２４－４参照）を、ＣＰＵモジュール１２とメモリモジュール１４－１とのデータ授受のために利用するとともに、同時に、メモリモジュール１４－２とメモリモジュール１４－３との間のデータ授受のために利用することができる（この場合に、スイッチ３０－５をオフにすれば良い）。したがって、より有効にバスを利用することが可能となってい

る。しかしながら、バスの組を数を十分に大きくできる場合、或いは、メモリモジュールの数が比較的少ない場合には、スイッチ２９或いは３０を必ずしも設けなくて良い。

【０１０３】また、本明細書において、制御信号ライン２５を介して、ＣＰＵ１２からのインストラクションが与えられることを記載したが、制御信号ライン２５を介して、インストラクションのほか、クロックなど、各メモリモジュールが同期して作動するための種々の制御信号が与えられ、かつ、各メモリモジュールからＣＰＵ１２への所定の信号（たとえば、エラー信号や、データ受理を示す信号）が与えられていることは言うまでもない。

【０１０４】さらに、前記実施の形態において、ＰＭＭ間の種々の接続を例示したが、ＰＭＭ間の接続や送受信に利用するバスの選択は、上記実施の形態に示すものに限定されない。

【０１０５】また、前記第３の実施の形態においては、図３２に示すように、第１のバス（符号３２０１）を利用して、各ＰＭＭ間の通信をなし、かつ、第２のバス（符号３２０２）参照）を利用して、要素や当該要素の存在数（重複数）が通信されているがこれに限定されるものではなく、たとえば、図４５に示すように、重複のない要素の配列である値リストやその存在数配列を生成するＰＭＭ“ｋ”が、第１のバス４５０１をモニターして、第１のバス４５０１上に表れる要素や存在数配列に基づき、所定の処理（たとえば、ＰＭＭ１～ＰＭＭ４にて実行されるカウンタのカウントアップやレジスタの内容の保持／更新）を実行して、値リストや存在数配列を作成しても良い。

【０１０６】さらに、本明細書において、一つの手段の機能が、二つ以上の物理的手段により実現されても、若しくは、二つ以上の手段の機能が、一つの物理的手段により実現されてもよい。

【０１０７】

【発明の効果】本発明によれば、著しく高速に、かつ、安定した処理時間で、配列のソート、コンパイルおよびジョインが可能な情報処理装置を提供することが可能となる。

【図面の簡単な説明】

【図１】　図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。

【図２】　図２は、本実施の形態にかかるメモリモジュールの概略を示すブロックダイヤグラムである。

【図３】　図３は、本実施の形態にかかるメモリモジュール間のパイプライン処理を説明するための図である。

【図４】　図４は、本実施の形態にかかる多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図５】　図５は、本実施の形態におけるメモリモジュールへのアクセスを説明するための図である。

【図６】　図６は、第１の実施の形態にかかるソート処理を施す配列の一例を示す図である。

【図７】　図７は、第１の実施の形態にかかるソート処理の処理手順を示すフローチャートである。

【図８】　図８は、第１の実施の形態にかかるソート処理を実施する際のメモリモジュール間の接続を示すブロックダイヤグラムである。

【図９】　図９は、図８に示すメモリモジュール間の接続を模式的に示す図である。

【図１０】　図１０は、第１の実施の形態にかかるソート処理における配列中の要素への番号付与を示す図である。

【図１１】　図１１は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１２】　図１２は、第１の実施の形態にかかるソート処理における配列中の要素への番号付与を示す図である。

【図１３】　図１３は、第１の実施の形態にかかるメモリモジュールのペア間の順位番号処理を示すフローチャートである。

【図１４】　図１４は、図８に示すメモリモジュールに関する、２つのメモリモジュール群の接続例を示すブロックダイヤグラムである。

【図１５】　図１５は、図１４に示す接続例を模式的に示す図である。

【図１６】　図１６は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１７】　図１７は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１８】　図１８は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１９】　図１９は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図２０】　図２０は、第１の実施の形態かかるソート処理におけるメモリモジュールの組み合わせを説明するための図である。

【図２１】　図２１は、第１の実施の形態にかかるソート処理の結果得られた順位番号にしたがって、新たな配列を生成する場合のメモリモジュールの接続例を示す図である。

【図２２】　図２２は、第１の実施の形態にかかるソート処理の結果得られた順位番号にしたがって、新たな配列を生成する場合のメモリモジュールの他の接続例を示

す図である。

【図２３】　図２３は、第２の実施の形態にかかるソート処理におけるメモリモジュールの接続例を模式的に示す図である。

【図２４】　図２４は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２５】　図２５は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２６】　図２６は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を示すフローチャートである。

【図２７】　図２７は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２８】　図２８は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２９】　図２９は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図３０】　図３０は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図３１】　図３１は、重複した要素の送出を排除したソート処理を示すフローチャートである。

【図３２】　図３２は、本発明の第３の実施の形態にかかるコンパイル処理におけるメモリモジュールの接続例を模式的に示す図である。

【図３３】　図３３は、第３の実施の形態にかかるコンパイル処理を示すフローチャートである。

【図３４】　図３４は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３５】　図３５は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３６】　図３６は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３７】　図３７は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３８】　図３８は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３９】　図３９は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図４０】　図４０は、本発明の第４の実施の形態にかかる共有化処理を示すフローチャートである。

【図４１】　図４１は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

【図４２】　図４２は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

【図４３】　図４３は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

【図４４】　図４４は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

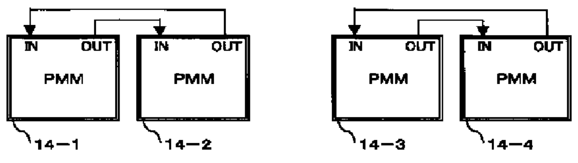
【図４５】　図４５は、本発明の他の応用例におけるメモリモジュール群の接続を概略的に示す図である。

【符号の説明】

１０	コンピュータシステム
１２	ＣＰＵモジュール
１４	メモリモジュール
１６	固定記憶装置
１８	入力装置
２０	表示装置
２２	レガシーメモリ
２４、２６	バス
２５	制御信号ライン
２８、２９、３０	スイッチ
３２	クロックバッファ
３４	ＲＡＭコア
３６	ＭＰＵ
３８	Ｉ／Ｏ

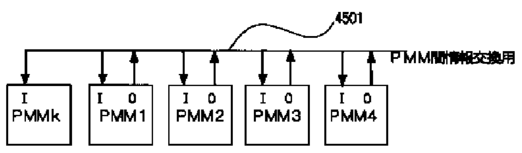
【図９】

図9



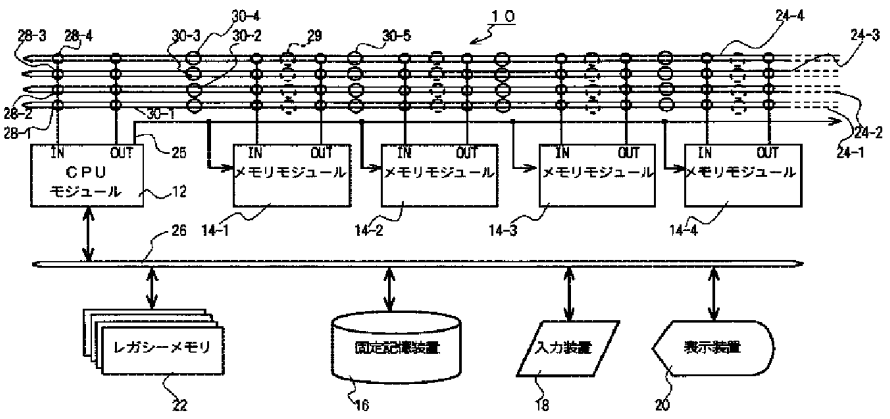
【図４５】

図４５



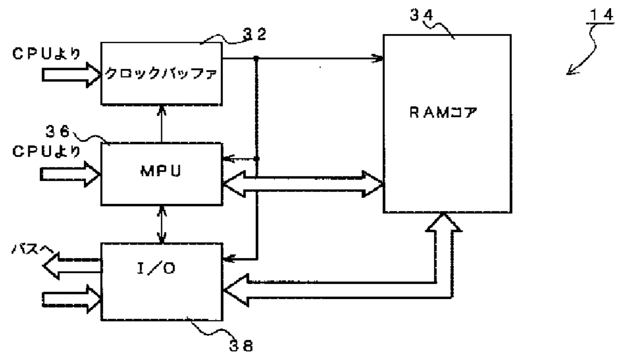
【図１】

図1



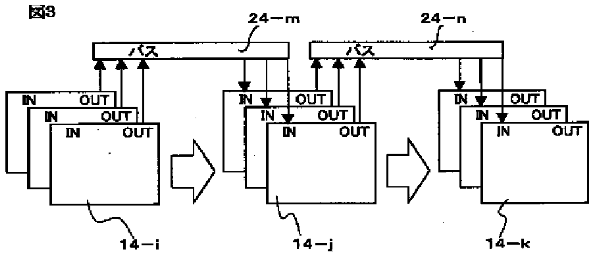
【図２】

図2



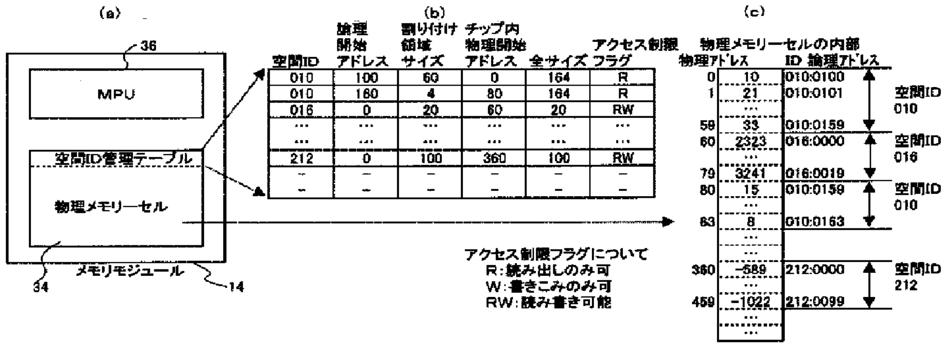
【図３】

図3



【図４】

図4



空間ID	論理開始アドレス	割り付け領域サイズ	チップ内物理開始アドレス	全サイズ	アクセス制限フラグ
010	100	60	0	164	R
010	160	4	80	164	R
016	0	20	60	20	RW
…	…	…	…	…	…
…	…	…	…	…	…
212	0	100	360	100	RW
－	－	－	－	－	－
－	－	－	－	－	－

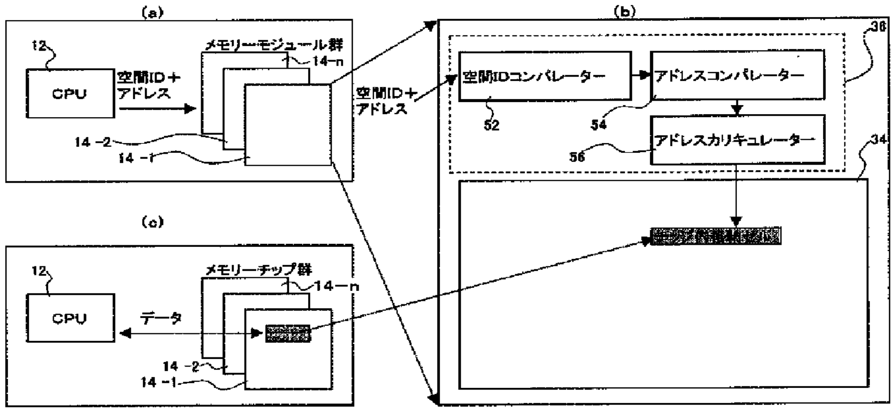
物理メモリーセルの内部

物理ｱﾄﾞﾚｽ		ID 論理ｱﾄﾞﾚｽ
0	10	010:0100
1	21	010:0101
	…	
59	33	010:0159
60	2323	016:0000
	…	
79	3241	016:0019
80	15	010:0159
	…	
83	8	010:0163
	…	
	…	
360	−589	212:0000
	…	
459	−1022	212:0099
	…	
	…	

アクセス制限フラグについて
R：読み出しのみ可
W：書きこみのみ可
RW：読み書き可能

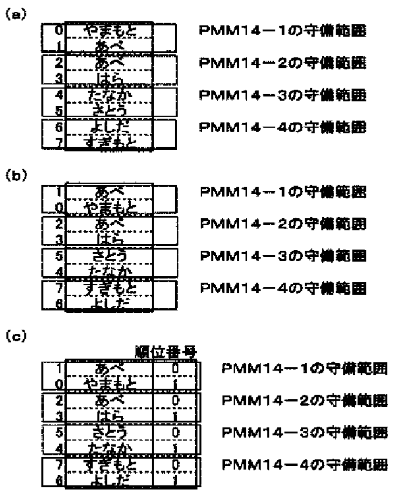
【図５】

図5



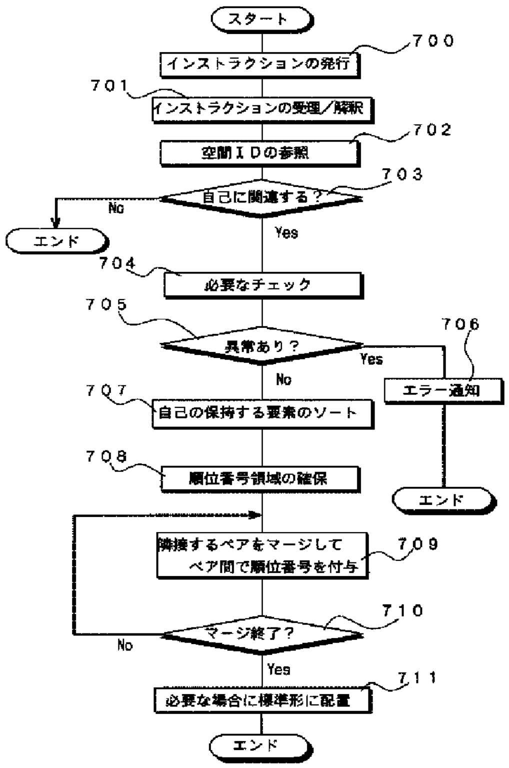
【図６】

図6

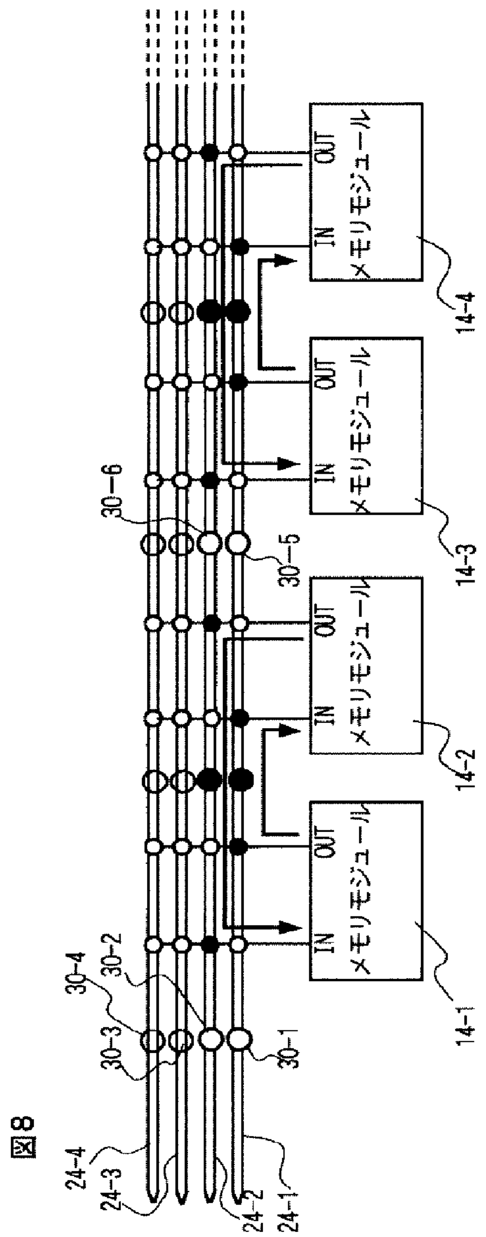


【図７】

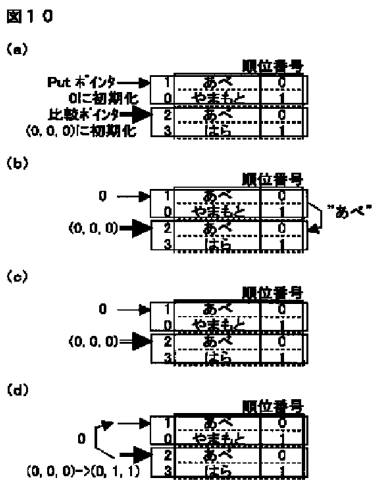
図７



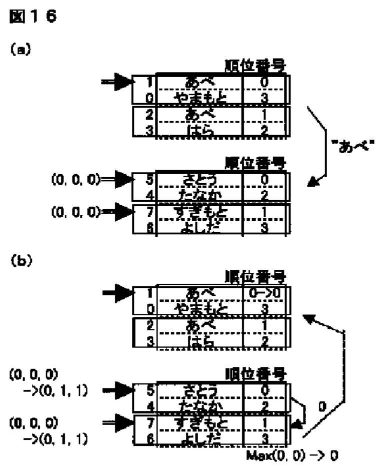
【図８】



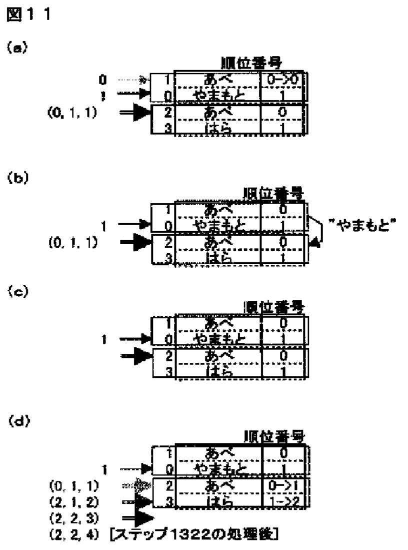
【図１０】



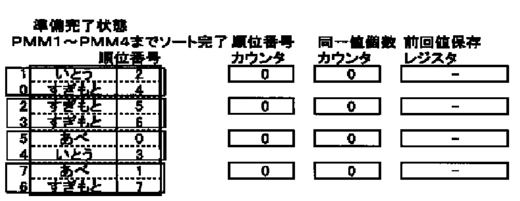
【図１６】



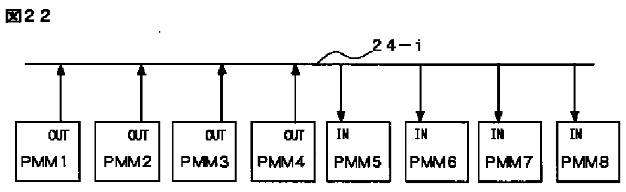
【図１１】



【図２５】

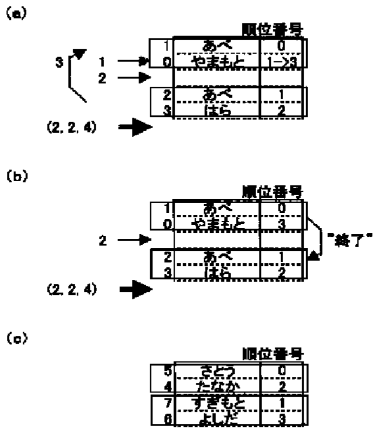


【図２２】



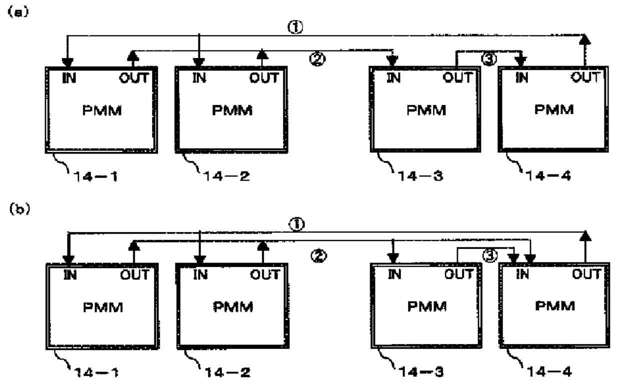
【図１２】

図１２



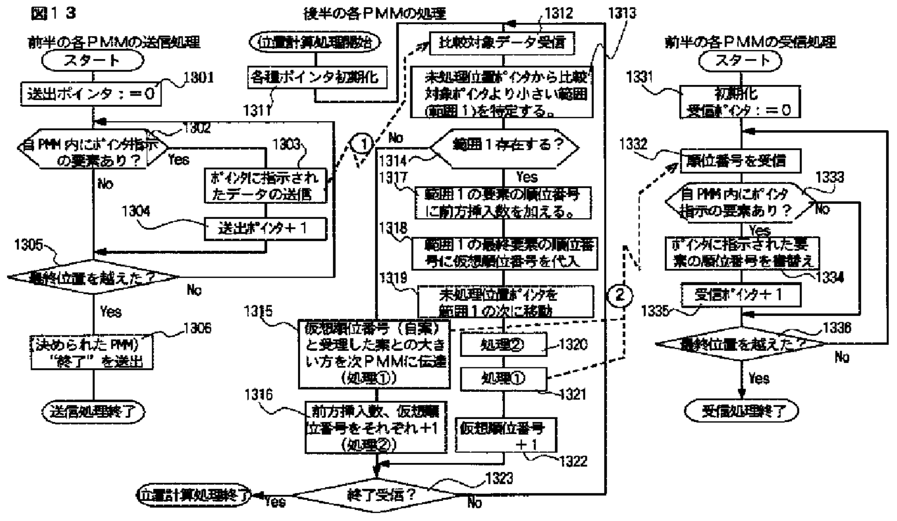
【図１５】

図１５



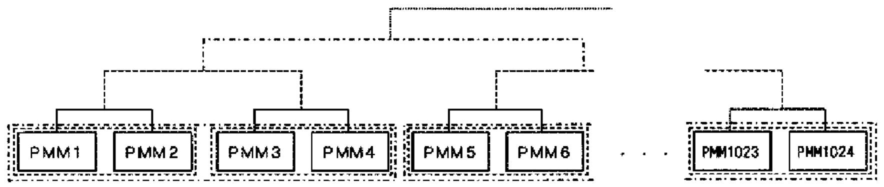
【図１３】

図１３


前半の各ＰＭＭの送信処理
スタート
送出ポインタ：＝0
1301
自PMM内にポインタ指示の要素あり？
1302
Yes
No
1303
ポインタに指示されたデータの送信
1304
送出ポインタ＋1
1305
最終位置を越えた？
No
Yes
（決められたPMM）"終了"を送出
1306
送信処理終了
後半の各ＰＭＭの処理
位置計算処理開始
各種ポインタ初期化
1311
1312
1313
比較対象データ受信
未処理位置ポインタから比較対象ポインタより小さい範囲（範囲１）を特定する。
①
No
範囲１存在する？
1314
Yes
1317
範囲１の要素の順位番号に前方挿入数を加える。
1318
範囲１の最終要素の順位番号に仮想順位番号を代入
1319
未処理位置ポインタを範囲１の次に移動
1315
仮想順位番号（自案）と受理した案との大きい方を次ＰＭＭに伝達（処理①）
1316
前方挿入数、仮想順位番号をそれぞれ＋1（処理②）
処理②
1320
処理①
1321
仮想順位番号＋1
1322
②
1323
終了受信？
Yes
No
位置計算処理終了
前半の各ＰＭＭの受信処理
スタート
1331
初期化 受信ポインタ：＝0
1332
順位番号を受信
1333
自PMM内にポインタ指示の要素あり？
No
Yes
ポインタに指示された要素の順位番号を書替え
1334
受信ポインタ＋1
1335
1336
最終位置を越えた？
No
Yes
受信処理終了

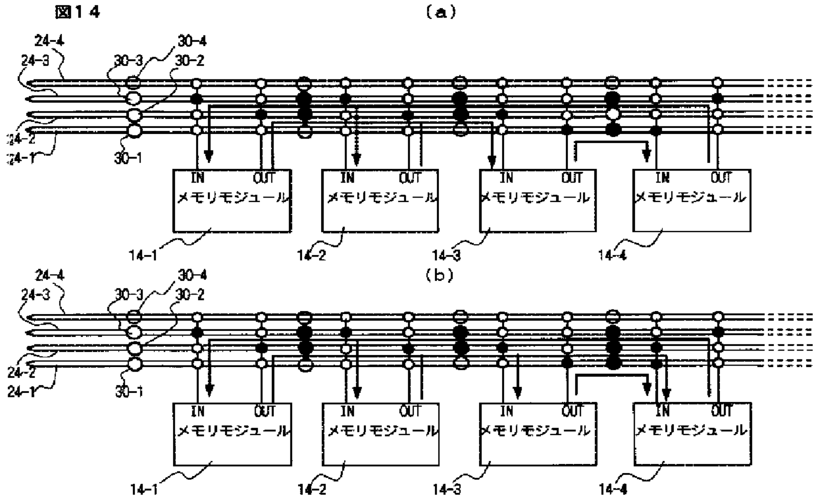
【図２０】

図２０



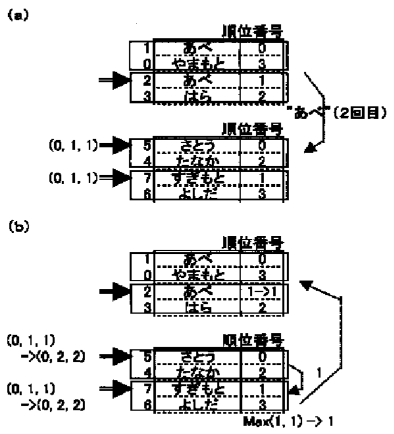
【図１４】

図１４



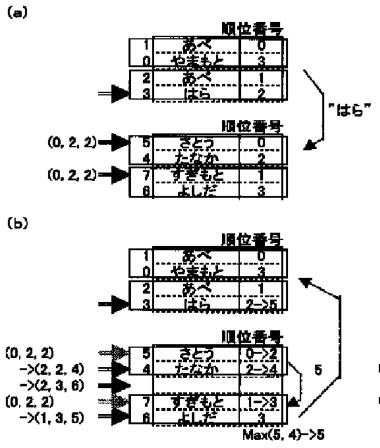
【図１７】

図１７



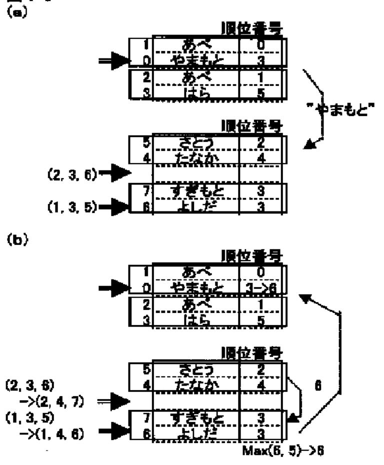
【図１８】

図１８



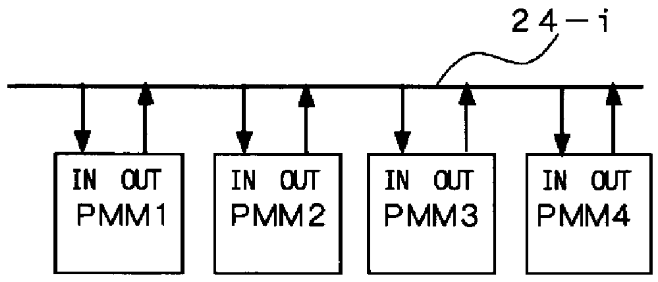
【図１９】

図１９



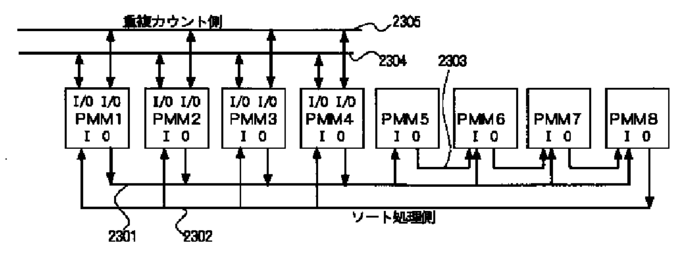
【図２１】

図２１



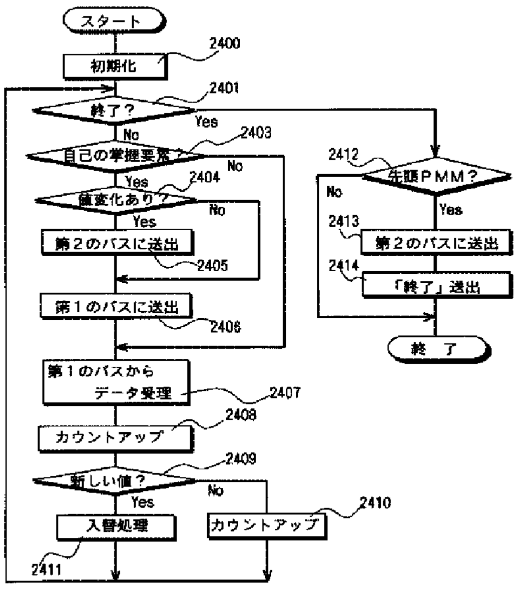
【図２３】

図２３



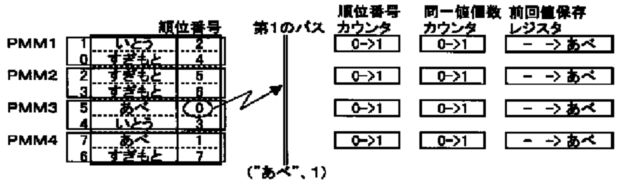
【図２４】

図２４



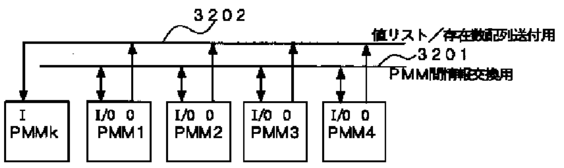
【図２６】

図２６



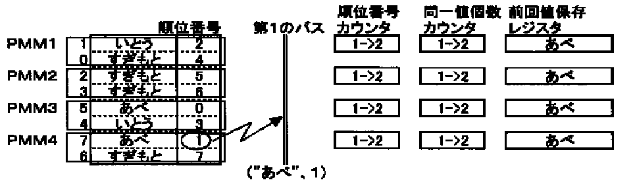
【図３２】

図３２



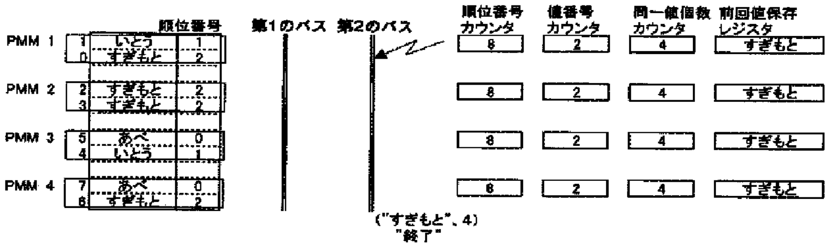
【図２７】

図２７



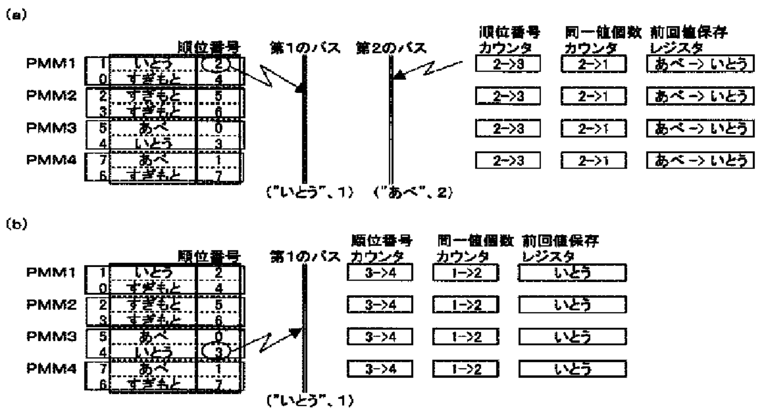
【図３８】

図38



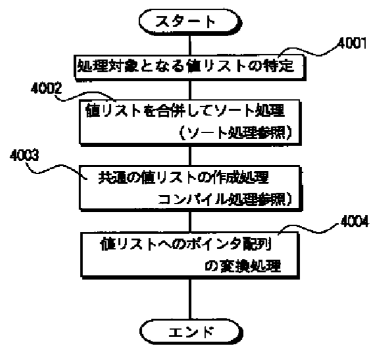
【図２８】

図２８



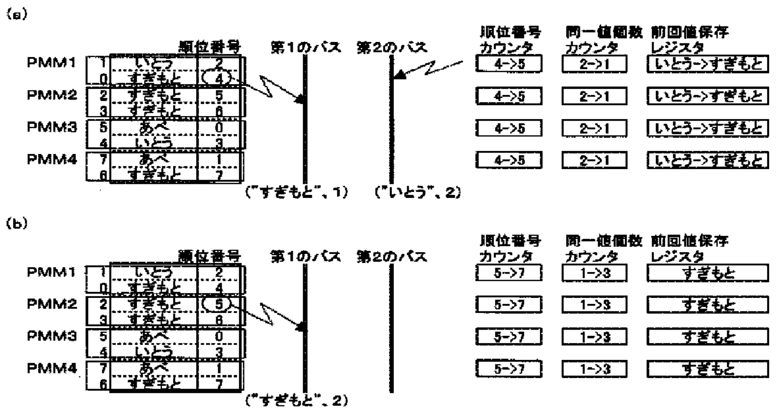
【図４０】

図４０


スタート
処理対象となる値リストの特定
4001
4002
値リストを合併してソート処理（ソート処理参照）
4003
共通の値リストの作成処理（コンパイル処理参照）
4004
値リストへのポインタ配列の変換処理
エンド

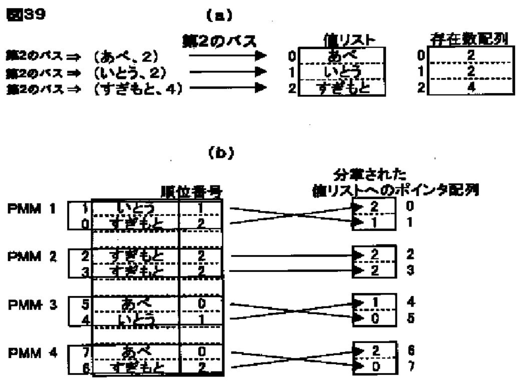
【図２９】

図２９



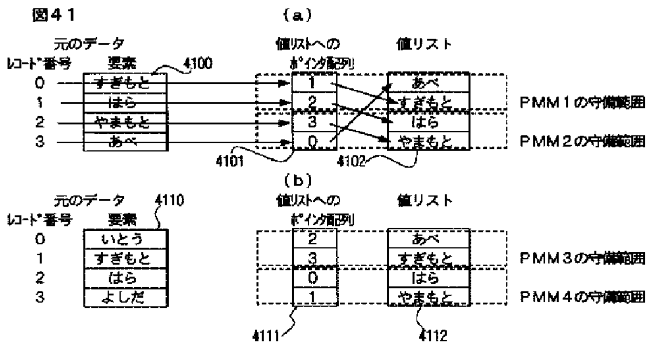
【図３９】

図39



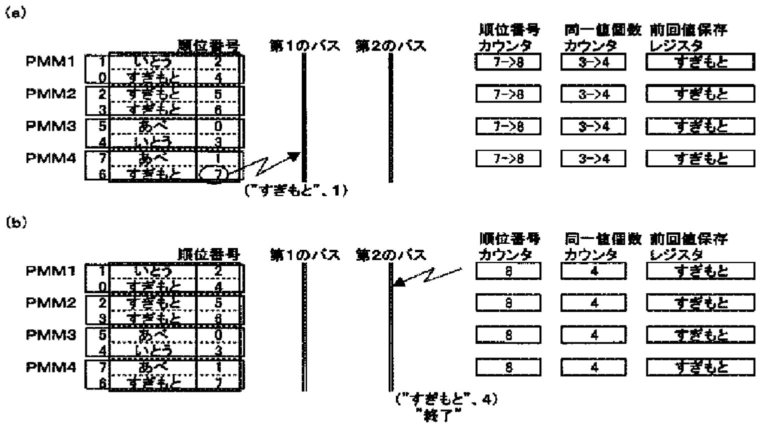
【図４１】

図４１



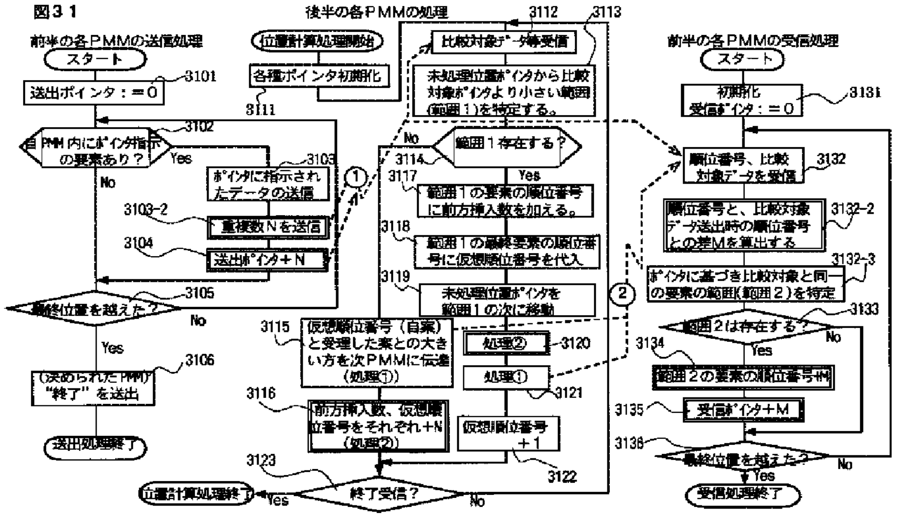
【図３０】

図３０



【図３１】

図３１


前半の各ＰＭＭの送信処理
スタート
送出ポインタ：＝０
3101
自PMM内にﾎﾟｲﾝﾀ指示の要素あり？
3102
Yes
No
ﾎﾟｲﾝﾀに指示されたデータの送信
3103
重複数Ｎを送信
3103-2
送出ﾎﾟｲﾝﾀ＋Ｎ
3104
最終位置を越えた？
3105
(決められたPMM)"終了"を送出
3106
送出処理終了
後半の各ＰＭＭの処理
位置計算処理開始
各種ポインタ初期化
3111
比較対象ﾃﾞｰﾀ等受信
3112
未処理位置ﾎﾟｲﾝﾀから比較対象ﾎﾟｲﾝﾀより小さい範囲(範囲１)を特定する。
3113
範囲１存在する？
3114
範囲１の要素の順位番号に前方挿入数を加える。
3117
範囲１の最終要素の順位番号に仮想順位番号を代入
3118
未処理位置ﾎﾟｲﾝﾀを範囲１の次に移動
3119
仮想順位番号（自案）と受理した案との大きい方を次ＰＭＭに伝達（処理①）
3115
前方挿入数、仮想順位番号をそれぞれ＋N（処理②）
3116
処理②
3120
処理①
3121
仮想順位番号＋１
3122
終了受信？
3123
位置計算処理終了
①
②
前半の各ＰＭＭの受信処理
スタート
初期化 受信ﾎﾟｲﾝﾀ：＝０
3131
順位番号、比較対象ﾃﾞｰﾀを受信
3132
順位番号と、比較対象ﾃﾞｰﾀ送出時の順位番号との差Ｍを算出する
3132-2
ﾎﾟｲﾝﾀに基づき比較対象と同一の要素の範囲(範囲２)を特定
3132-3
範囲２は存在する？
3133
範囲２の要素の順位番号+M
3134
受信ﾎﾟｲﾝﾀ＋Ｍ
3135
最終位置を越えた？
3136
受信処理終了

【図３３】

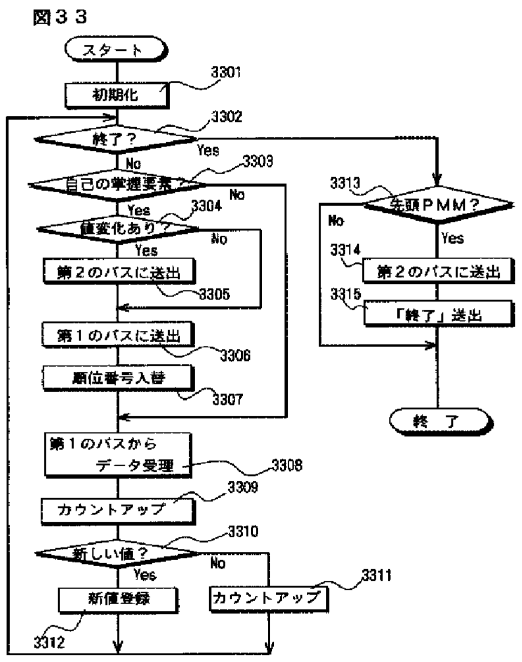


図３３
スタート
初期化
3301
終了？
3302
Yes
No
自己の掌握要素？
3303
No
Yes
値変化あり？
3304
No
Yes
第２のバスに送出
3305
第１のバスに送出
3306
順位番号入替
3307
第１のバスから
データ受理
3308
カウントアップ
3309
新しい値？
3310
Yes
No
新値登録
3312
カウントアップ
3311
先頭ＰＭＭ？
3313
No
Yes
第２のバスに送出
3314
「終了」送出
3315
終　了

【図４２】

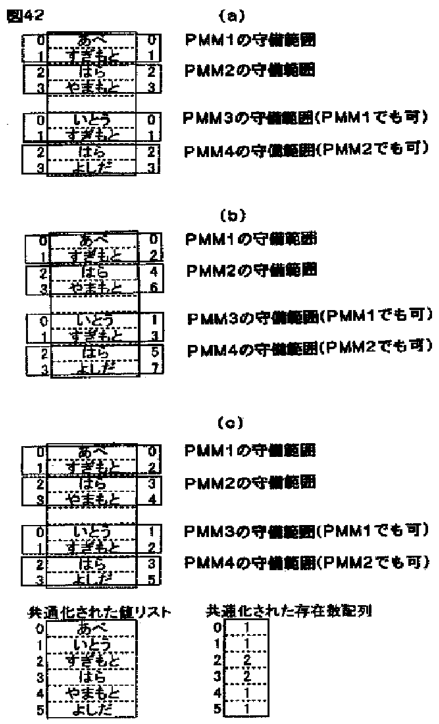


図42
(a)
あべ
すぎもと
はら
やまもと
いとう
すぎもと
はら
よしだ
PMM1の守備範囲
PMM2の守備範囲
PMM3の守備範囲(PMM1でも可)
PMM4の守備範囲(PMM2でも可)
(b)
PMM1の守備範囲
PMM2の守備範囲
PMM3の守備範囲(PMM1でも可)
PMM4の守備範囲(PMM2でも可)
(c)
PMM1の守備範囲
PMM2の守備範囲
PMM3の守備範囲(PMM1でも可)
PMM4の守備範囲(PMM2でも可)
共通化された値リスト
あべ
いとう
すぎもと
はら
やまもと
よしだ
共通化された存在数配列

【図３４】

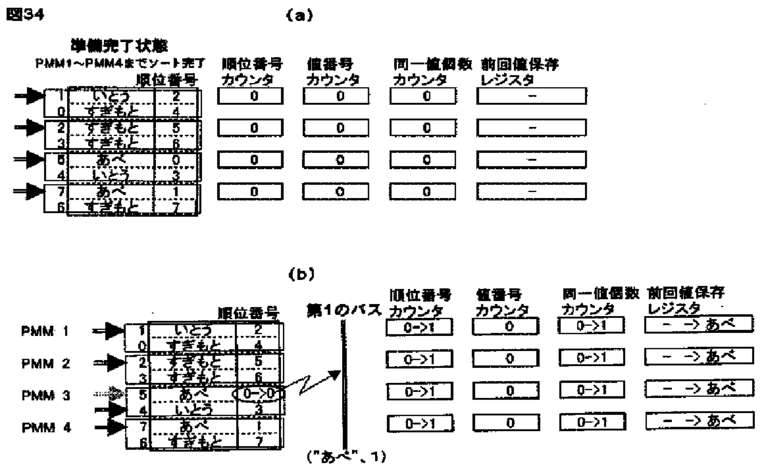


図34
(a)
準備完了状態
PMM1～PMM4までソート完了
順位番号
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
いとう
すぎもと
すぎもと
すぎもと
あべ
いとう
あべ
すぎもと
(b)
PMM 1
PMM 2
PMM 3
PMM 4
順位番号
第1のバス
0->1
0->0
－ -> あべ
("あべ"、1)

【図３５】

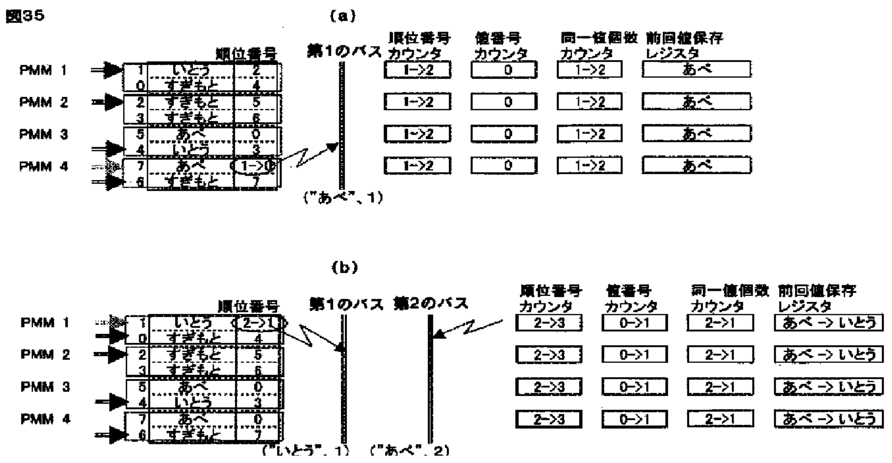


図35
(a)
(b)
PMM 1
PMM 2
PMM 3
PMM 4
順位番号
第1のバス
第2のバス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
いとう
すぎもと
あべ
1->2
1->0
2->1
2->3
0->1
あべ -> いとう
("あべ"、1)
("いとう"、1)
("あべ"、2)

【図３６】

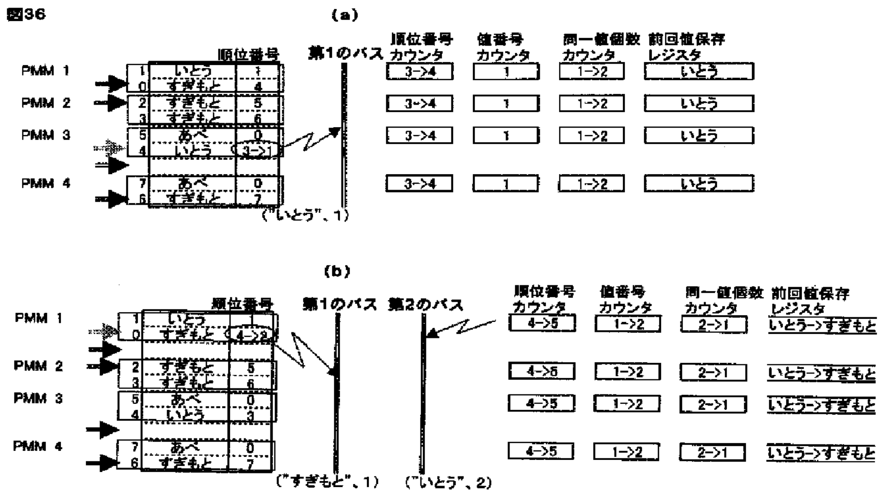
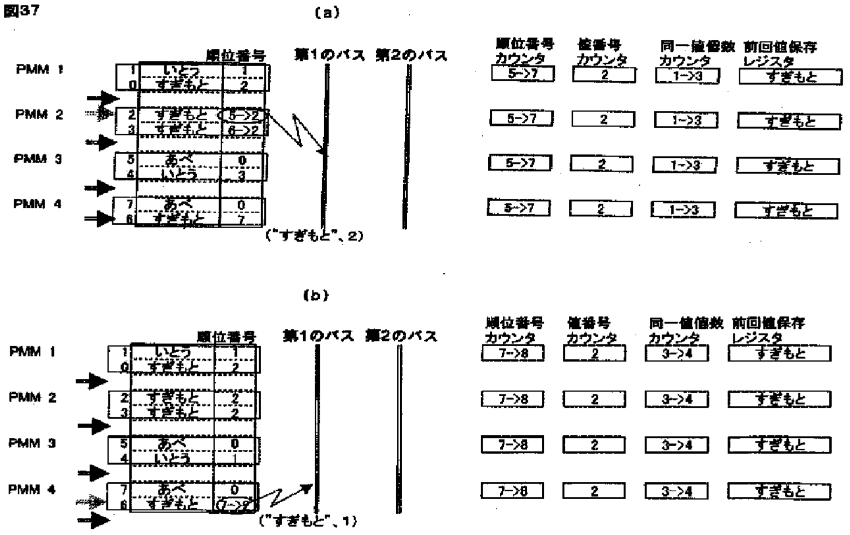


図36
(a)
(b)
PMM 1
PMM 2
PMM 3
PMM 4
順位番号
第1のバス
第2のバス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
いとう
すぎもと
あべ
3->4
3->1
1->2
4->5
4->2
2->1
いとう->すぎもと
("いとう"、1)
("すぎもと"、1)
("いとう"、2)

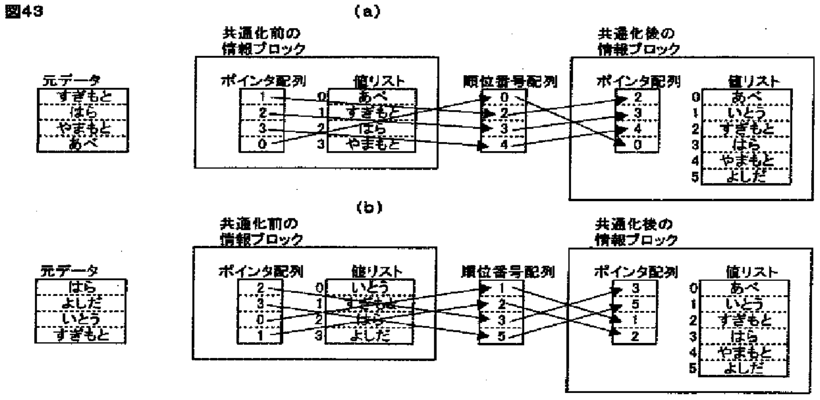
【図３７】

図37

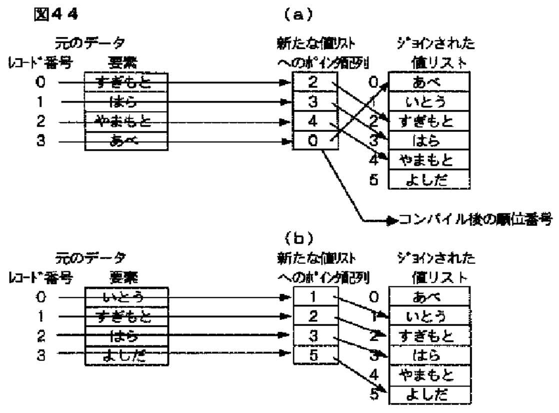


【図４３】

図43



【図４４】



フロントページの続き

(51)Int.Cl.^{7}	識別記号	F I	テーマコード (参考)
		Ｇ０６Ｆ　15/403	３４０Ｄ